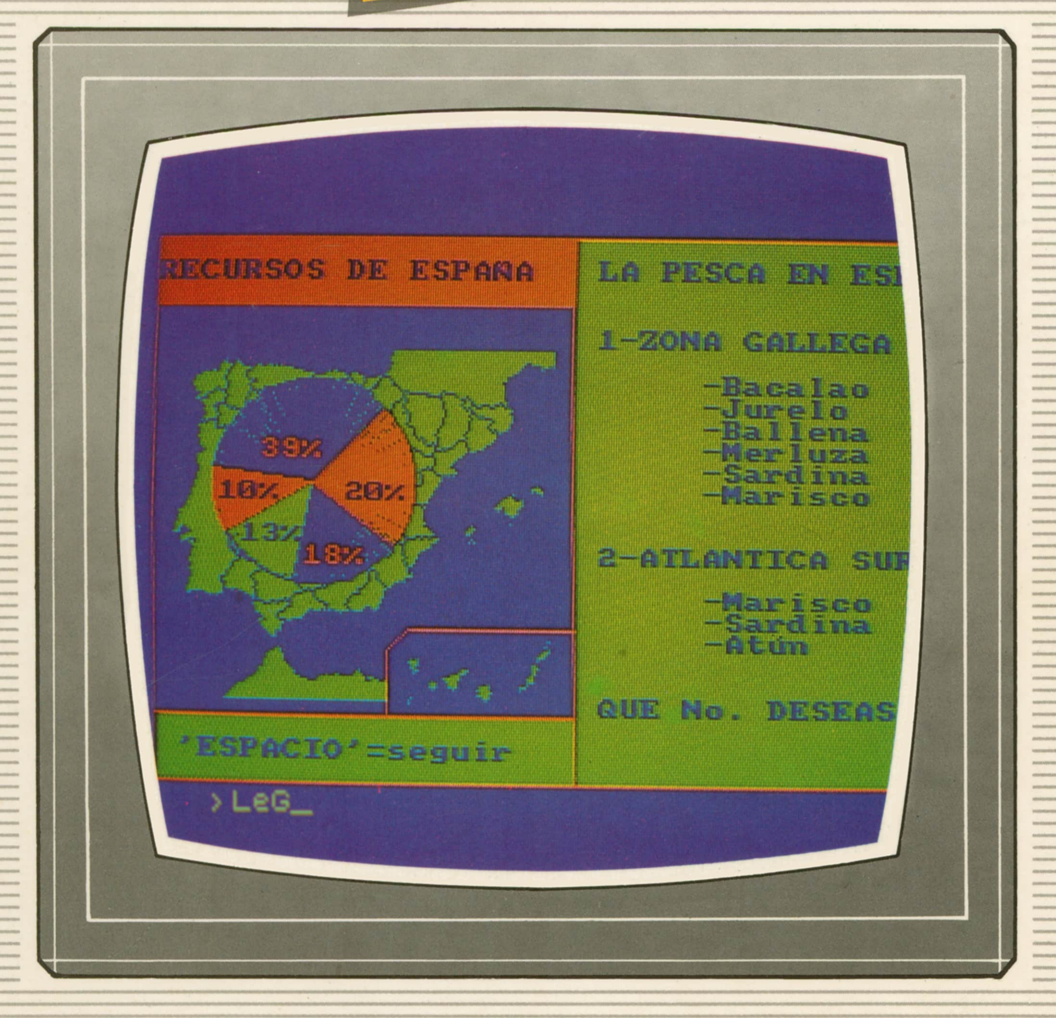
▼ AMSTRAD ▼ SPECTRUM ▼ COMMODORE ▼ IBM ▼ MSX ▼

Informática y programación 16

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Informática y programación 16

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Una publicación de

EDICIONES SIGLO CULTURAL, S.A.

Director-editor:
RICARDO ESPAÑOL CRESPO.

Gerente:
ANTONIO G. CUERPO.

Directora de producción:
MARIA LUISA SUAREZ PEREZ.

Directores de la colección:
MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación y Licenciado en Informática.
JOSE ARTECHE, Ingeniero de Telecomunicación.

Diseño y maquetación:
BRAVO-LOFISH.

Fotografía:
EQUIPO GALATA.

Dibujos:
JOSE OCHOA

TECNICAS DE PROGRAMACION: Manuel Alfonseca, Doctor Ingeniero de Telecomunicación y Licenciado en Informática. TECNICAS DE ANALISIS: José Arteche, Ingeniero en Telecomunicación. LENGUAJE MAQUINA 8086: Juan Rojas Licenciado en Ciencias Físicas e Ingeniero Industrial. PASCAL: Juan Ignacio Puyol, Ingeniero Industrial. PROGRAMAS (educativos, de utilidad, de gestión y de juegos): Francisco Morales, Técnico en Informática y colaboradores. Coordinador de AULA DE INFORMATICA APLICADA (AIA): Alejandro Marcos, Licenciado en Ciencias Químicas. BASIC: Esther Maldonado, Diplomada en Arquitectura. INFORMATICA BASICA: Virginia Muñoz, Diplomada en Informática. LENGUAJE MAQUINA Z-80: Joaquín Salvachúa, Diplomado en Telecomunicación y José Luis Tojo, Diplomado en Telecomunicación. LENGUAJE MAQUINA 6502: (desde el tomo 5): Juan José Gómez, Licenciado en Química. LOGO: Cristina Manzanera, Licenciada en Informática. APLICACIONES: Sociedad Tamariz, Diplomada en Telecomunicación. OTROS LENGUAJES (COBOL): Eloy Pérez, Licenciado en Informática. Ana Pastor, Licenciada en Informática.

Ediciones Siglo Cultural, S.A.

Dirección, redacción y administración:
Pedro Teixeira, 8, 2.ª planta. Teléf. 810 52 13. 28020 Madrid.

Publicidad:
Gofar Publicidad, S.A. Benito de Castro, 12 bis. 28028 Madrid.

Distribución en España:
COEDIS, S.A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-126-6
ISBN de la obra: 84-7688-068-7

Fotocomposición:
ARTECOMP, S.A. Albarracín, 50. 28037 Madrid.

Imprime:
MATEU CROMO. Pinto (Madrid).

Depósito legal: M-5-677-1987
Printed in Spain - Impreso en España.

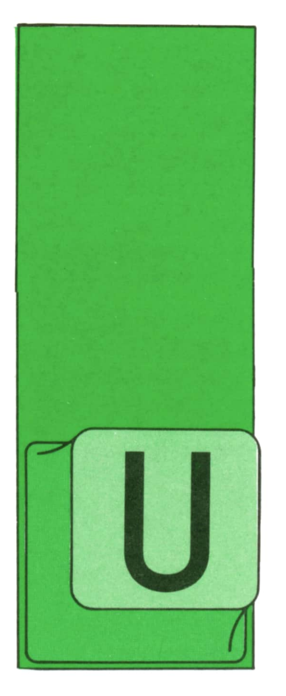
Suscripciones y números atrasados:
Ediciones Siglo Cultural, S.A.
Pedro Teixeira, 8, 2.ª planta. Teléf. 810 52 13. 28020 Madrid.

Julio, 1987.
P.V.P. Canarias: 335,–.

INDICE

INFORMATICA BASICA

LAS BASES DE DATOS

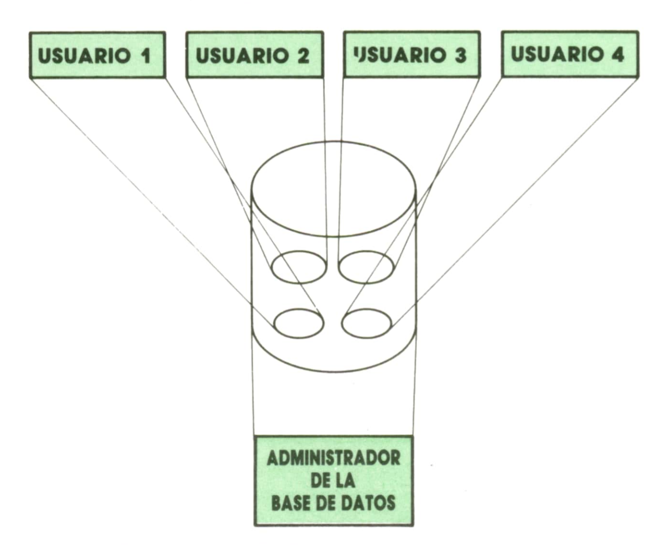


NA **base de datos** se puede definir como un conjunto de datos interrelacionados que cumplen una serie de requisitos tecnológicos en cuanto a su diseño lógico y a la organización de los datos en los soportes físicos, lo que las distingue de los ficheros.



Orígenes

Inicialmente con la monoprogramación, las aplicaciones que utilizaban ficheros de datos solían tener sus propios datos, y en pocas ocasiones estos datos eran compartidos por otros programas. Al parecer la multiprogramación y producirse un gran desarrollo en los sistemas operativos, se permitió que varias aplicaciones se ejecutasen a un tiempo, surgiendo así nuevos problemas.

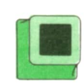


Cada uno de los usuarios de la base de datos va a tener solamente una visión parcial de la base. El encargado de definir cada una de estas «visiones» es el administrador.

En una organización clásica de ficheros, un mismo conjunto de datos se encuentra en varios ficheros; los datos se recogen varias veces en distintos archivos. Esta redundancia, además de malgastar recursos, origina a menudo divergencias en los resultados, produce una ocupación inútil de memoria secundaria y un aumento de los tiempos de proceso. Además, debido a que los datos se encuentran repetidos en más de un fichero, la actualización no se realiza de forma simultánea provocando incoherencias.

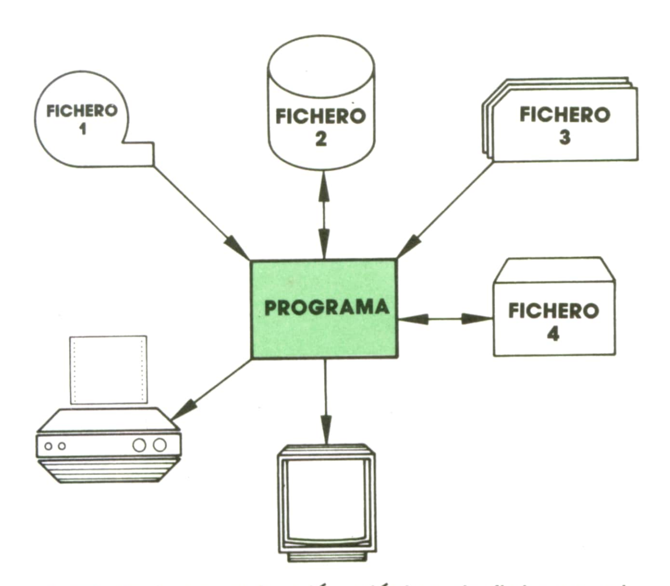
Esta duplicación de informaciones es lo que se conoce con el nombre de **redundancia** y las incoherencias provocadas por tales redundancias producen lo que se llama **inconsistencia** de la información.

Por otro lado, en la organización clásica de ficheros, la dependencia de los datos respecto a las aplicaciones es muy fuerte, provocando la reprogramación de dichas aplicaciones cuando se producen cambios en los datos.



Aparición de las bases de datos

El nacimiento de las bases de datos viene acelerado, precisamente por la aparición de la multiprogramación y todos los problemas que se derivan de la utilización de los ficheros. Se empieza a replantear la recogida y organización de los datos, de forma que constituyan un conjunto estructurado no sólo válido para una aplicación concreta, sino para satisfacer las necesidades de información global del usuario.




En la organización clásica de ficheros, el programa y los ficheros están íntimamente relacionados.

Por tanto, podemos hacer una enumeración de las características principales que reúnen las bases de datos:

— **Independencia de los datos y programas.** Para implementar esta independencia es necesario hacer replanteamiento del concepto de información, analizando con más detalle su concepto y los métodos utilizados para representarla.

La información consiste en acumular ideas y hechos acerca de diferentes cosas: personajes, lugares, herramientas, etcétera. Estas **cosas** se denominan entidades.




Pasos en la recogida de informaciones.

Es decir, se registra información sobre entidades que consta de:

— **Contexto.** Define el ámbito de la entidad, es decir, el contexto es el mismo en entidades iguales, y diferente, si son distintas.

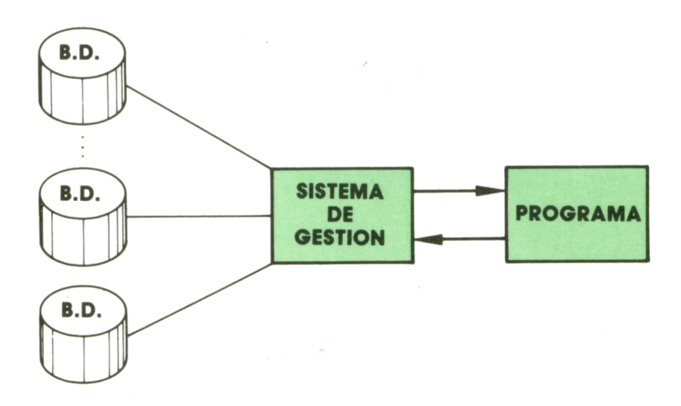
— **Datos.** Marcan unos valores concretos para variables del ámbito de la entidad.

— **Representación de los datos.** Indica el formato sobre el cual se representan los datos.

En la práctica esta independencia nunca es absoluta, aunque se puede llegar a un nivel óptimo a un coste elevado.

— **Coherencia de los resultados.** Con la creación de las bases de datos, al unificar todos los ficheros en uno único, se evitan duplicaciones y redundancias. Por otro lado, al no estar repetida la información, las actualizaciones no producen inconsistencias. La integridad de los datos se facilitará al realizar todas las operaciones a través de un gestor. Las lecturas o escritura no se realizan directamente desde los programas de las aplicaciones, sino que se realizan mediante un sistema de gestión de la base.

— **Se reduce el espacio de almacenamiento.** No sólo al evitar las redundancias, sino también por la utilización de técnicas de comprensión de la información. Esto hace que aumente el volumen informativo (el valor informativo de la base de datos es superior a la suma del valor informativo de los elementos que la constituyen al estar éstos interrelacionados).



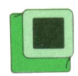
Los accesos a la base de datos se realizan a través del sistema de gestión, que es quien se encarga de la transferencia de informaciones entre los distintos programas y la base de datos.

No sólo existen ventajas en la utilización de las bases de datos, los principales inconvenientes de su implantación son:

— Se requiere mayor protección de datos, ya que usuarios distintos deben acceder al mismo fichero. Las bases de datos tienen que proveer de mayores controles de seguridad a la información.
— Instalaciones costosas. Tanto en material (ordenadores potentes y dispositivos de memoria secundaria con gran capacidad de almacenamiento) como en logical.
— Personal especializado, difícil de encontrar, pero fácil de perder.
— Implementación larga y difícil.
— Rentabilidad a medio plazo.

Una vez conocidas todas estas características podremos definir más exactamente una base de datos como:

«Un conjunto de datos relacionados entre sí lógicamente con una definición común, almacenada junto con los datos, y con una cierta estructuración que permite su utilización por parte de múltiples usuarios.



Operaciones típicas de una base de datos

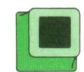
En una base de datos podremos hacer operaciones:

— Sobre el conjunto de la base:

— Creación.
— Reestructuración.
— Consulta de la totalidad.

— Sobre registros aislados:

— Inserción de datos.
— Borrado.
— Modificaciones.
— Consulta selectiva.

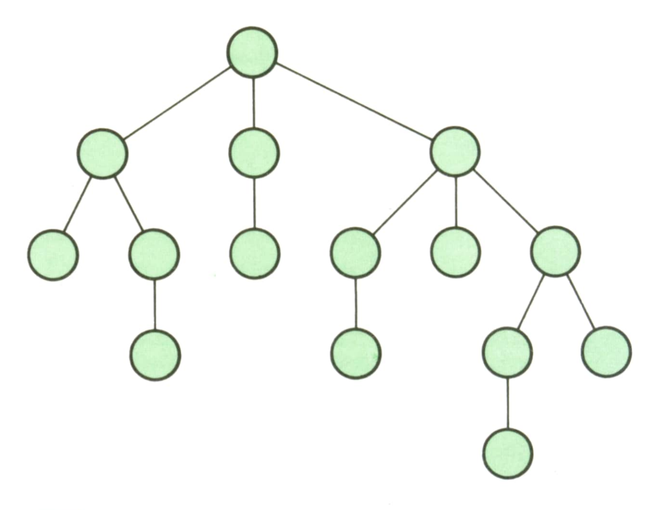


Estructura de una base de datos

Se distinguen la **estructura lógica** de la **estructura física.** La primera es la que el usuario define y por la que se rige para efectuar las consultas. La segunda es la forma en la que se encuentran los datos en el almacenamiento; es mucho más compleja que la otra debido a que se deben incluir los índices, encadenamientos, etc.

Los tres modelos más conocidos son:

— **Modelo jerárquico**




Representación gráfica de una base de datos de tipo jerárquico.

Tiene una estructura arborescente. Los distintos nodos muestran los elementos de la base de datos que, como sabemos, se denominan **entidades.** Cada nodo de nivel superior puede tener varios nodos relacionados a él en un nivel inferior (hijos), pero sólo pueden estar relacionados con un nodo de nivel superior (padre).

— **Modelo relacional**

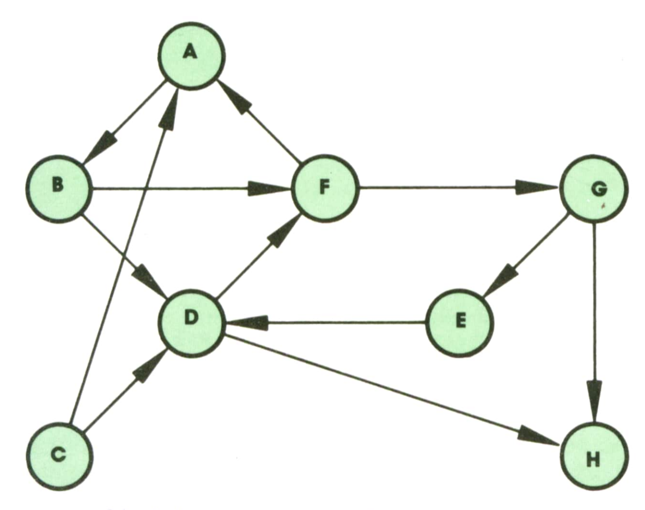
Nombre Atributo 1	D.N.I. Atributo 2	Teléfono Atributo 3
JUAN	43.175.203	228 25 33
EDUARDO	9.741.662	77 94 15
ENRIQUE	15.275.105	23 06 72
⋮	⋮	⋮
AGUSTIN	9.745.835	222 66 30


Modelo relacional. Representación en forma de tabla.

El concepto básico de este modelo es «la relación» entre entidades. Se repre-

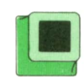
sentan mediante unas tablas en las que las columnas son «los atributos», es decir, las propiedades o características de las entidades, y las filas, los valores que tienen dichos atributos.

— **Modelo en red**




Modelo en red. Los distintos nodos indican las entidades y las flechas representan las asociaciones o relaciones entre ellas.

Es la combinación de varias jerarquías arborescentes, en la que un «hijo» puede tener varios «padres». Este modelo es mucho más flexible que los otros. Se representa mediante conjuntos de anillos con un registro propietario y varios registros miembros, que pueden ser del mismo o distinto tipo.



El sistema de gestión de base de datos

Un sistema de gestión de base de datos es un conjunto coordinado de programas, procedimientos, lenguajes, etc., que suministra a los usuarios, informáticos o no, los medios necesarios para poder describir, manipular y recuperar los datos integrados en la base, asegurando su confidencialidad y seguridad.

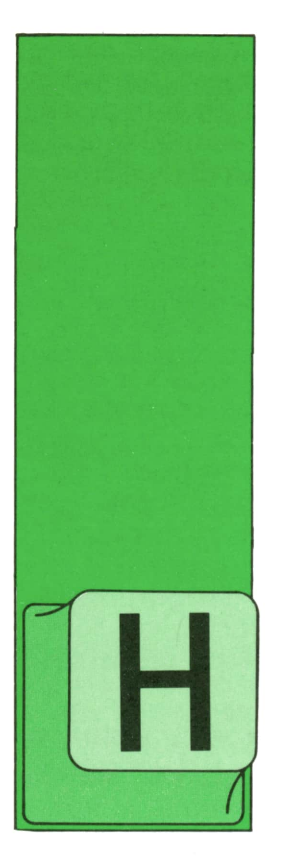
Las funciones esenciales de un sistema de gestión son:

— Describir los elementos de datos, su estructura, sus relaciones y sus validaciones.

El sistema de gestión de bases de datos utiliza diferentes tipos de lenguajes para manipular la base de datos.

MAQUINA 8088

LA PILA DE EJECUCION



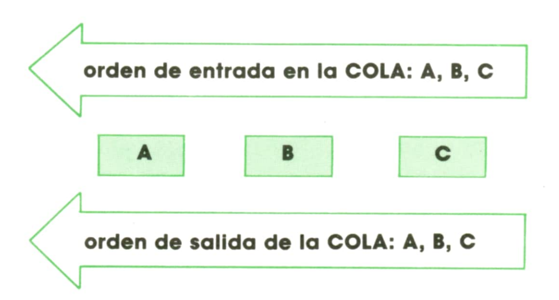
HEMOS visto anteriormente que el microprocesador 8088, maneja las cuatro áreas de memoria siguientes:

— Area de código.
— Area de datos origen.
— Area de datos destino.
— Area de la pila de ejecución.

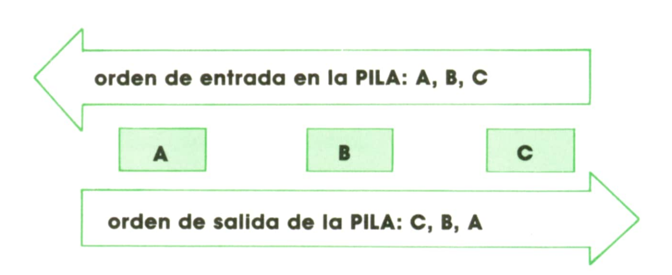
Pues bien, es de esta última área de la que vamos a tratar a continuación.

Empezaremos por definir el concepto de «pila» y diferenciarlo del concepto de «cola». Nos ayudaremos para ello de los siguientes ejemplos:

Imaginemos la situación que se produce cuando llegan a un peaje por el que tienen que pasar de uno en uno, más vehículos de los que se puede atender. En ese caso los vehículos se disponen formando una «cola». Las colas funcionan con la regla: «el primero en llegar es el primero en salir». A esta regla se la conoce como regla FIFO («first-in first-out»).



Supongamos ahora el caso en que una serie de vehículos embarca en un ferry estrecho y largo con una puerta de embarque única. A la hora de desembarcar no se puede usar la regla anterior porque cada vehículo que llega impide la salida de todos los que llegaron antes que él. En este caso hay que emplear la regla «el último en llegar es el primero en salir», que se conoce con el hombre LIFO («last-in first-out»). A la disposición de los vehículos dentro del ferry, como a todas las que presenten estas características, se las denomina «pila».



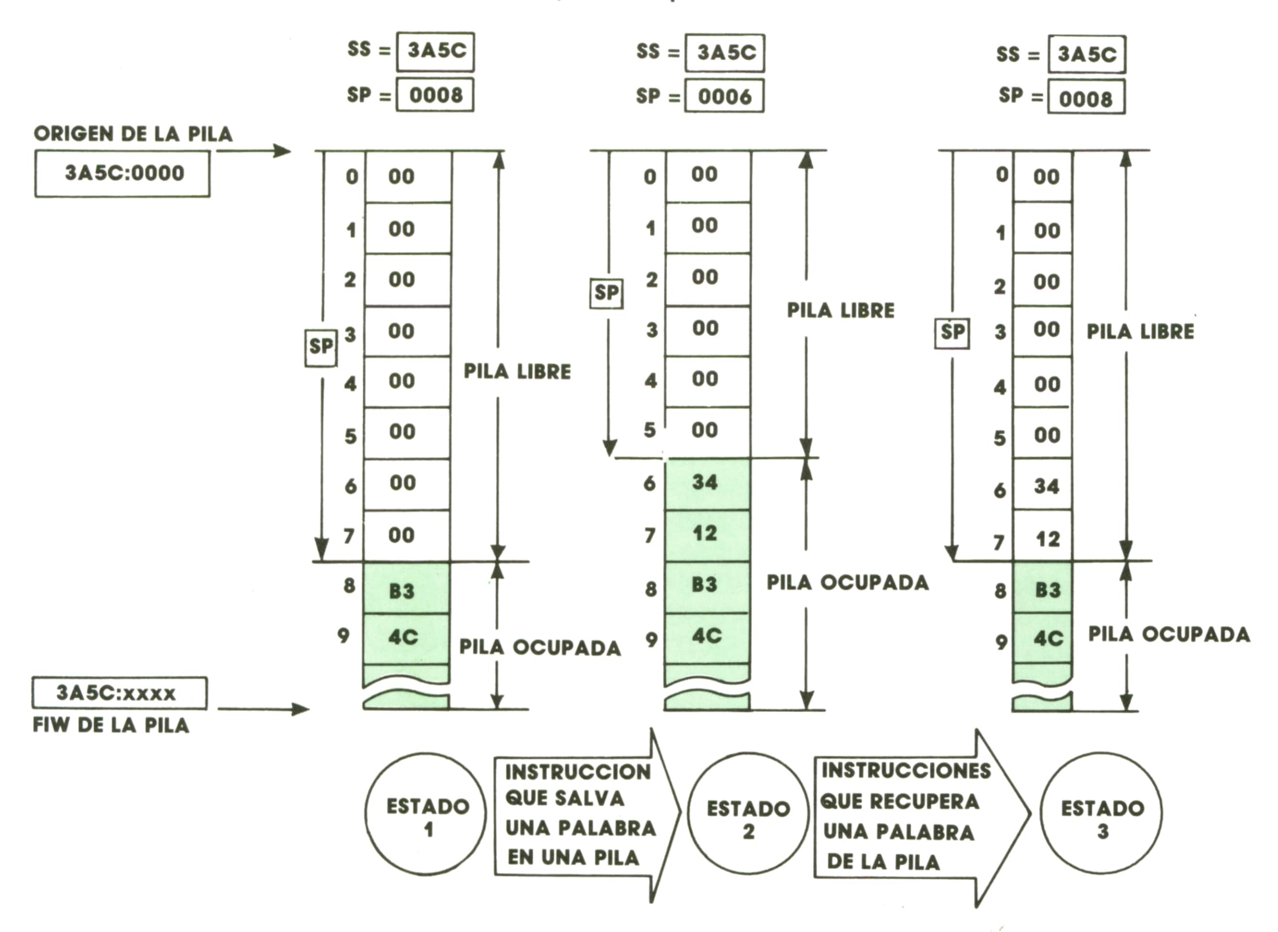
El 8088 controla directamente, de acuerdo con la regla LIFO, un área de memoria que recibe el nombre de pila de ejecución o STACK. Este área es una característica muy importante de la programación con el 8088, que facilita la escritura de programas recurrentes, es decir, de programas que pueden llamarse a sí mismos un determinado número de veces.

El área de memoria stack está direccionada como ya se dijo, por la pareja de registros SS y SP. Esto quiere decir que el microprocesador calcula la dirección física de memoria tomando el SS como re-

gistro de segmento y el SP como registro de desplazamiento.

En la programación normal, las instrucciones que usan la pila se ejecutan siempre por parejas. Primero se ejecutan las instrucciones que «salvan» información en la pila y después se ejecutan instrucciones que «recuperan» dicha información. La forma más normal de programar es dividir el programa principal en módulos elementales (llamados rutinas). Cada módulo usa la pila salvando y recuperando la misma cantidad de información, de forma que al terminar la ejecución de cada módulo la pila se encuentre en la misma situación que cuando empezó.

En la siguiente figura se muestran los estados de la pila en tres instantes diferentes. El estado 1 es el estado en que se encuentra la pila en un momento determinado. Supongamos que en ese instante el registro SS contenga 3A5C que indica la situación de la pila en la memoria y que el registro SP contenga el valor 0008 que apunta al último byte ocupado de la pila.



Si se ejecuta una instrucción que «salva» una palabra (dos bytes) en la pila se pasa al estado 2. La transición se realiza en dos pasos: primero se decrementa en dos unidades el registro SP (que pasa a tomar el valor 0006) y después se escribe la palabra que se desea guardar (supongamos que sea 12 34) en la dirección definida por SS:SP.

Si a partir de esta situación se ejecuta una instrucción que «recupera» una palabra de la pila, se pasa al estado 3. Este tipo de instrucciones se realiza también en dos pasos: primero se recupera la palabra apuntada por SS:SP (en nuestro caso 12 34) y después se incrementa el registro SP en dos unidades, pasando así a valer 0008.

Hay varias instrucciones que «salvan» información en la pila y otras tantas que «recuperan» la información previamente salvada. Estas instrucciones, que en la programación aparecen normalmente emparejadas, son las siguientes:

— La pareja PUSH y POP.
— La pareja PUSHF y POPF.
— La pareja CALL y RET.
— La pareja INT e IRET.

En algunos casos se «salva» una palabra en la pila, como en las instrucciones PUSH, PUSHF y CALL tipo «near»; en las instrucciones CALL tipo «far» se «salvan» dos palabras y en las INT se «salvan» tres palabras.

En las instrucciones asociadas se «recupera» la misma cantidad de información. Es decir, una palabra en POP, POPF y RET tipo «near»; dos palabras en las instrucciones RET tipo «far» y tres palabras en las instrucciones IRET.

Los mecanismos por los que se «salvan» y «recuperan» dos o tres palabras son los mismos que los explicados para una palabra, pero repetidos dos o tres veces en la instrucción.

A continuación vamos a explicar con detalle estas instrucciones.



Instrucciones PUSH y POP

Son dos instrucciones independientes que vamos a tratar simultáneamente, ya que realizan funciones una inversa de la otra y en los programas se emplean normalmente emparejadas.

La instrucción PUSH sirve para «salvar» (es decir, guardar) una palabra en la pila. El funcionamiento es el siguiente: primero se decrementa en dos unidades el registro SP y después se guarda la palabra en la dirección referenciada por SS:SP.

La instrucción POP sirve para «recuperar» (es decir extraer) una palabra de la pila. La palabra se extrae de la dirección referenciada por SS:SP y después se incrementa en el registro SP en dos unidades.

Estas instrucciones tienen los formatos siguientes:

```
[etiqueta]   PUSH   origen
[etiqueta]   POP    destino
```

Recordemos que en éste, como en los demás formatos que vamos a utilizar, se emplean los siguientes convenios: lo escrito con minúsculas son nombres que deben ser elegidos por el programador; lo escrito con mayúsculas son nombres que deben ser codificados como se indica; y lo encerrado entre corchetes son elementos opcionales que pueden ser omitidos.

Las etiquetas son opcionales como en todas las instrucciones ejecutables y se pueden emplear para dar un nombre a las instrucciones que vayan a ser referenciadas desde otras instrucciones del programa.

PUSH y POP son los códigos de operación.

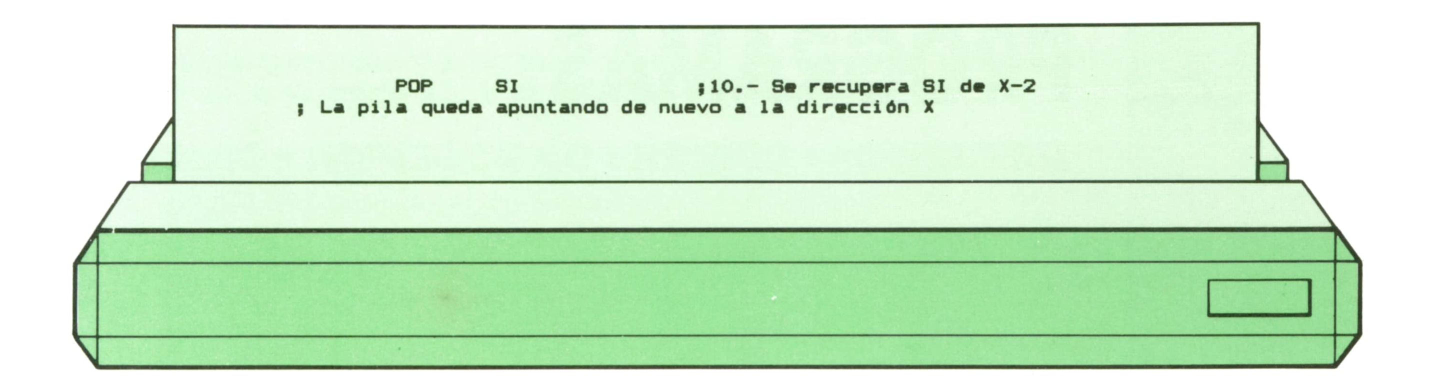
Y «origen» y «destino» son los operandos asociados a estas dos instrucciones. El operando «origen» de la instrucción PUSH define de dónde se debe extraer la palabra que se desea «salvar» en la pila. El operando «destino» de la instrucción POP define dónde se debe guardar la palabra que se quiere «recuperar» de la pila.

Estos operandos pueden ser:

— Registros (AX, BX, CX, DX, SI, DI, SP o BP).
— Registros de segmento (DS, ES, SS o CS) excepto el CS en la instrucción POP.
— Referencias explícitas a la memoria en cualquiera de las formas que se han explicado.

Veamos un trozo de programa que usa instrucciones PUSH y POP:

```
; LLamemos X, para abreviar, a la dirección actual de la pila SS:SP
        PUSH     SI                  ; 1.- Se salva SI en X-2
        PUSH     AX                  ; 2.- Se salva AX en X-4
        PUSH     DS                  ; 3.- Se salva DS en X-6
        PUSH     ALFA [SI] [BX]      ; 4.- Se salva una palabra en X-8
        PUSH     DI                  ; 5.- Se salva DI en X-10
        ...
; La pila apunta ahora a la dirección X-10
; Se ejecutan ahora instrucciones intermedias que no alteran la pila.
        ...
        POP      DI                  ; 6.- Se recupera DI de X-10
        POP      BETA [DI] [BP]      ; 7.- Se recupera una palabra de X-8
        POP      ES                  ; 8.- Se recupera ES de X-6
        POP      BX                  ; 9.- Se recupera BX de X-4
```



```
        POP     SI                  ;10.- Se recupera SI de X-2
; La pila queda apuntando de nuevo a la dirección X
```

En este ejemplo, la palabra contenida en el registro AX se recupera en el registro BX. La palabra contenida en el registro DS termina transferida al registro ES. La palabra contenida en la dirección de memoria ALFA (SI) (DI) acaba en la dirección de memoria BETA (DI) (BP). Por último los registros SI y DI recuperan sus valores aunque en las instrucciones intermedias se hubieran alterado los mismos.



Instrucciones PUSHF y POPF

Estas dos instrucciones realizan funciones una inversa de la otra.

La instrucción PUSHF sirve para «salvar» en la pila la palabra de estado de los flags. Antes de salvarlos decrementa en dos unidades el registo SP y después guarda los flags en la palabra direccionada por SS:SP.

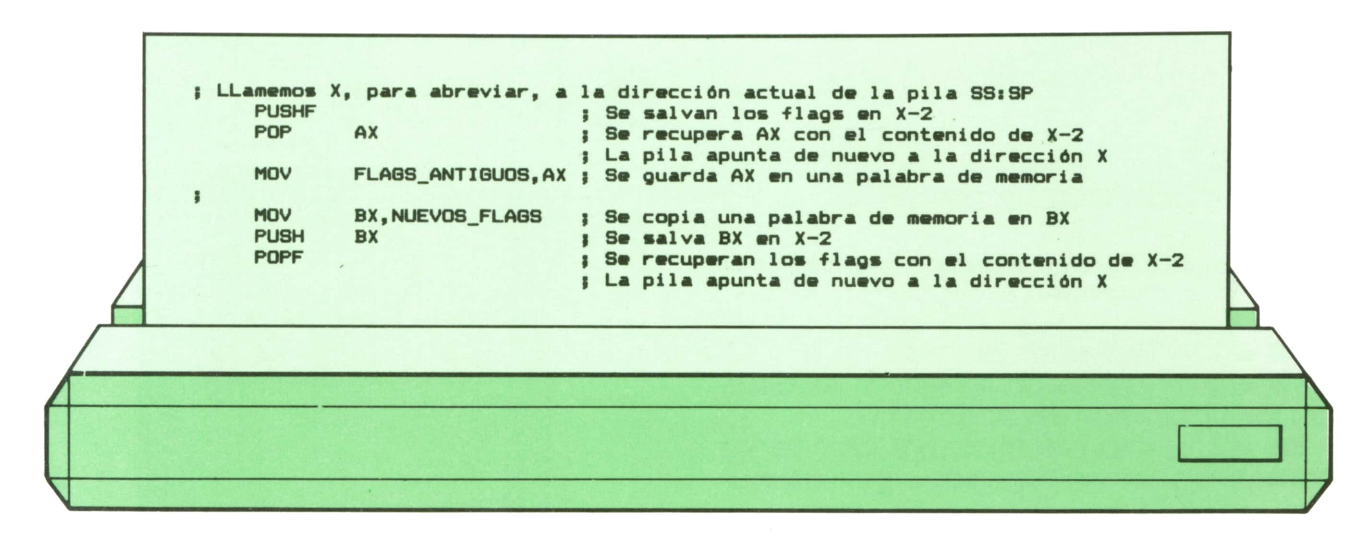
La instrucción POPF sirve para «recuperar» de la pila una determinada configuración de los flags. Se extraen de la palabra direccionada por SS:SP y después se incrementa el registro SP en dos unidades.

Los formatos de estas instrucciones son los siguientes:

```
[etiqueta]   PUSHF
[etiqueta]   POPF
```

La etiqueta es opcional. PUSHF y POPF son los códigos de operación. Y como se observa, son instrucciones que no llevan operandos.

En el siguiente ejemplo se dan una serie de instrucciones que permiten copiar los flags activos en una palabra de memoria (denominada aquí FLAGS-ANTIGUOS) y definir un nuevo juego de flags a partir de otra palabra de memoria (denominada aquí NUEVOS-FLAGS):

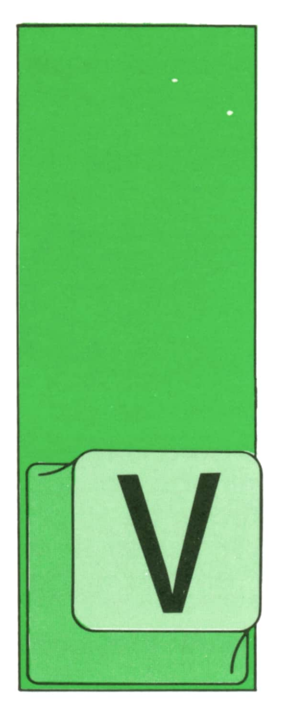


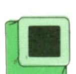
```
; LLamemos X, para abreviar, a la dirección actual de la pila SS:SP
        PUSHF                           ; Se salvan los flags en X-2
        POP     AX                      ; Se recupera AX con el contenido de X-2
                                        ; La pila apunta de nuevo a la dirección X
        MOV     FLAGS_ANTIGUOS,AX       ; Se guarda AX en una palabra de memoria
;
        MOV     BX,NUEVOS_FLAGS         ; Se copia una palabra de memoria en BX
        PUSH    BX                      ; Se salva BX en X-2
        POPF                            ; Se recuperan los flags con el contenido de X-2
                                        ; La pila apunta de nuevo a la dirección X
```

Obsérvese que se han emparejado por un lado las instrucciones PUSHF y POP y por otro lado las instrucciones PUSH y POPF.

PROGRAMAS

EDUCATIVOS • DE UTILIDAD • DE GESTION • DE JUEGOS



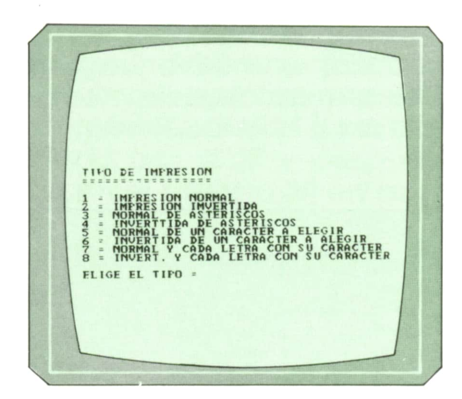


Programa: letras gigantes

VAMOS a ver a continuación un programa que nos permitirá imprimir caracteres ocho veces más grandes de lo normal. Este programa utiliza su propio juego de caracteres, distinto del de la máquina en la que se instale.

Aunque el programa está escrito en BASIC, es lo suficientemente rápido como para poderlo utilizar como una rutina en nuestros programas.

Los mensajes que este programa nos permite imprimir pueden ser de varios tipos, dependiendo del valor que le demos a la variable SW. Estos son:





Todos los tipos de impresión que soporta el programa.

SW = 1 — Impresión normal.

SW = 2 — Impresión invertida. Se dibujan los contornos de cada letra.

SW = 3 — Impresión normal, donde los caracteres están formados por asteriscos.

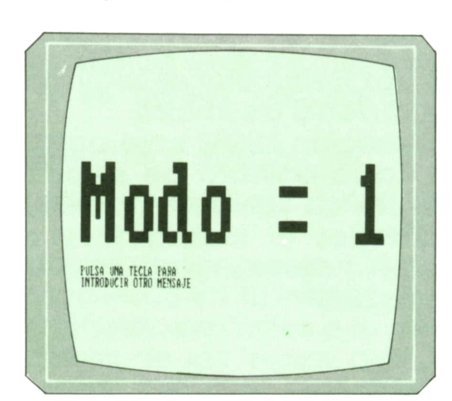
SW = 4 — Impresión invertida, donde los contornos de los caracteres están formados por asteriscos.

SW = 5 —Impresión normal, donde los caracteres están formados por un carácter que nosotros elijamos y que tiene que estar almacenado en L$.

SW = 6 —Impresión invertida, donde los contornos de cada carácter están formados por un carácter que nosotros elijamos y que ha de estar almacenado en L$.

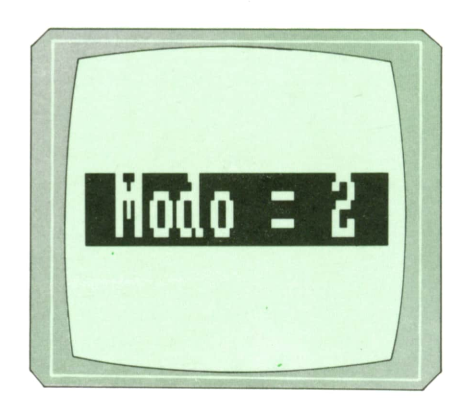
SW = 7 — Impresión normal, donde cada letra está formada por el mismo carácter que intenta representar.

SW = 8 — Impresión invertida, donde el contorno de cada letra está formado por letras que son las mismas que la letra que se intenta representar.





Ejemplo del Modo 1.





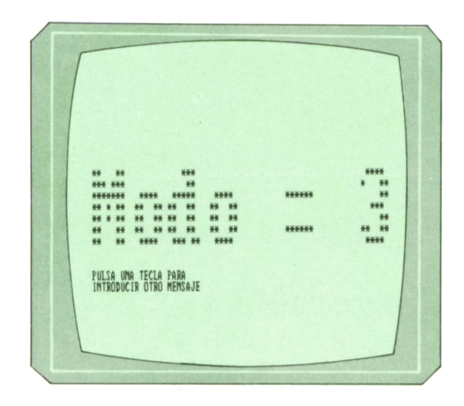
Ejemplo del Modo 2.

El mensaje a representar ha de ir almacenado en la variable A$, y la posición del mensaje en la pantalla, en las coordenadas X e Y.

```
8000 REM *************************************************************
8010 REM *                                                           *
8020 REM *         L      EEEEEE TTTTTT RRRRR   AAAA   SSSS          *
8030 REM *         L      E    E T TT T R    R A    A S    S         *
8040 REM *         L      E  E     TT   R    R A    A S              *
8050 REM *         L      EEEE     TT   RRRRR  AAAAAA  SSSS          *
8060 REM *         L      E  E     TT   RRR    A    A      S         *
8070 REM *         L      E    E   TT   R RR   A    A S    S         *
8080 REM *         LLLLLL EEEEEE. TTTT  R  RR  A    A  SSSS          *
8090 REM *                                                           *
8100 REM *   GGGG  IIIIII  GGGG   AAAA  N    N TTTTTT EEEEEE  SSSS   *
8110 REM *  G    G   II   G    G A    A NN   N T TT T E    E S    S  *
8120 REM *  G        II   G      A    A N N  N   TT   E  E   S       *
8130 REM *  G  GGG   II   G  GGG AAAAAA N NN N   TT   EEEE    SSSS   *
8140 REM *  G    G   II   G    G A    A N  N N   TT   E  E        S  *
8150 REM *  G    G   II   G    G A    A N   NN   TT   E    E S    S  *
8160 REM *   GGGG  IIIIII  GGGG  A    A N    N  TTTT  EEEEEE  SSSS   *
8170 REM *                                                           *
8180 REM *************************************************************
8190 REM
8200 REM *****************************************
8210 REM * LECTURA DE LAS DATAS Y ALMACENAMIENTO *
8220 REM *****************************************
8230 REM
8240 DEF FNZ=16*(INSTR(Z$,MID$(B$,1,1))-1)+INSTR(Z$,MID$(B$,2,1))-1
8250 LET Z$="0123456789ABCDEF"
8260 DIM N$(24)
8270 LET T=0
8280 LET L=9070
8290 FOR I=1 TO 24
8300    READ A$,A
8310    FOR J=1 TO LEN(A$) STEP 2
8320       LET B$=MID$(A$,J,2)
8330       LET T=T+FNZ
8340    NEXT J
8350    IF T<>A THEN PRINT "ERROR EN LA LINEA ";L:END
8360    LET T=0
8370    LET L=L+20
8380    LET N$(I)=A$
8390 NEXT I
8400  RETURN
8500 REM
8510 REM **********************
8520 REM * PROGRAMA PRINCIPAL *
8530 REM **********************
8540 REM
8550 IF SW<2 THEN LET M$=CHR$(219):LET W$=" "
8560 IF SW=2 THEN LET M$=" ":LET W$=CHR$(219)
8570 IF SW=3 THEN LET M$="*":LET W$=" "
8580 IF SW=4 THEN LET M$=" ":LET W$="*"
8590 IF SW=5 THEN LET M$=L$:LET W$=" "
8600 IF SW=6 THEN LET M$=" ":LET W$=L$
8610 FOR I=1 TO LEN(A$)
8620    LET N=ASC(MID$(A$,I,1))
8630    IF SW=7 THEN LET M$=CHR$(N):LET W$=" "
8640    IF SW>7 THEN LET M$=" ":LET W$=CHR$(N)
8650    LET N=N-31
8660    LET L=INT(N/4)+1
8670    LET P=N-4*(L-1)
8680    LET P=(P-1)*16+1
8690    IF P<0 THEN LET L=L-1:LET P=49
8700    FOR J=1 TO 8
8710       LOCATE Y+J-1,X
```

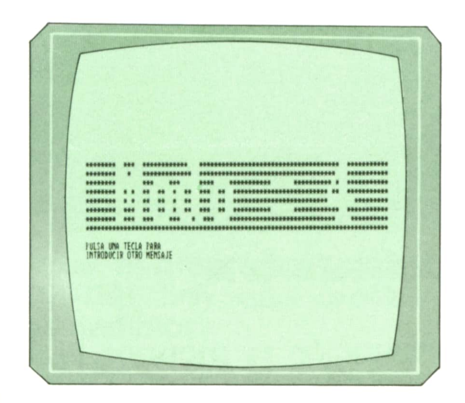
```
8720        LET B$=MID$(N$(L),P+2*J-2,2)
8730        LET A=FNZ
8740        IF A>=128 THEN LET A=A-128:PRINT M$;:GOTO 8760
8750        PRINT W$;
8760        IF A>=64 THEN LET A=A-64:PRINT M$;:GOTO 8780
8770        PRINT W$;
8780        IF A>=32 THEN LET A=A-32:PRINT M$;:GOTO 8800
8790        PRINT W$;
8800        IF A>=16 THEN LET A=A-16:PRINT M$;:GOTO 8820
8810        PRINT W$;
8820        IF A>=8 THEN LET A=A-8:PRINT M$;:GOTO 8840
8830        PRINT W$;
8840        IF A>=4 THEN LET A=A-4:PRINT M$;:GOTO 8860
8850        PRINT W$;
8860        IF A>=2 THEN LET A=A-2:PRINT M$;:GOTO 8880
8870        PRINT W$;
8880        IF A=1 THEN PRINT M$;:GOTO 8900
8890        PRINT W$;
8900        PRINT
8910     NEXT J
8920     LET X=X+8
8930     IF X>80 THEN LET X=1:LET Y=Y+8
8940 NEXT I
8950 RETURN
9000 REM
9010 REM ******************************
9020 REM * LINEAS DE DATAS CON NUMEROS *
9030 REM * EN HEXADECIMAL CON LA DEFI- *
9040 REM * NICION DE LOS CARACTERES.   *
9050 REM ******************************
9060 REM
9070 DATA "000000000000000018181818180018006C6C0000000000006C6CFE7CFE6C6C00"
9080 DATA 1532
9090 DATA "183E583C1A7C18000063660C18336300386C3876DCDC7600181830000000000"
9100 DATA 1787
9110 DATA "0C18303030180C0030180C0C0C18300000663CFF3C6600000018187E18180000"
9120 DATA 1197
9130 DATA "00000000001818300000007E0000000000000000001818000060C183060C08000"
9140 DATA 776
9150 DATA "7CC6CED6E6C67C0018381818181887E003C66063C60667E003C46061C06663C00"
9160 DATA 2480
9170 DATA "18385898FE183C007E62603C06663C003C66607C66663C007E46060C18181800"
9180 DATA 2138
9190 DATA "3C66663C66663C003C66663E06663C0000001818001818000000181800181830"
9200 DATA 1322
9210 DATA "0C18306030180C0000007E00007E00006030180C183060003C66060C18001800"
9220 DATA 1092
9230 DATA "7CC6DEDEDEC07C00183C66667E666600FC66667C6666FC003C66C0C0C0663C00"
9240 DATA 3858
9250 DATA "F86C6666666CF800FE6268786862FE00FE62687868 60F0003C66C0C0CEC67E00"
9260 DATA 4142
9270 DATA "6666667E666666007E18181818187E001E0C0C0CCCCC7800E6666C786C66E600"
9280 DATA 2704
9290 DATA "F0606060626 6FE00C6EEFED6D6C6C600C6E6F6DECEC6C600386CC6C6C66C3800"
9300 DATA 4916
9310 DATA "FC666678606 0F000386CC6C6DACC7600FC66667C6C66E2003C66603C06663C00"
9320 DATA 3610
9330 DATA "7E5A181818183C006666666666663C006666666666663C1800C6C6C6D6FEEEC600"
9340 DATA 3136
9350 DATA "C66C38386CC6C60066666663C18183C00FEC68C183266FE003C30303030303C00"
9360 DATA 2778
9370 DATA "C06030180C0603013C0C0C0C0C0C3C00183C7E18181818000000000000000000FF"
9380 DATA 1123
9390 DATA "0030180C00000000000780C7CCC7600E0E07C666666BC0000003C6660663C00"
9400 DATA 2148
9410 DATA "1C0C7CCCCCCC760000003C667E603C001C3630783030780000003E66663E067C"
9420 DATA 2262
9430 DATA "E0606C766666E60018003818181 83C0002000E060666663CE060666C786CE600"
```

```
9440 DATA 2472
9450 DATA "3818181818183C0000006CFED6D6C6000000D8666666660000003C6666663C00"
9460 DATA 2274
9470 DATA "0000DC66667C60F0000076CCCC7C0C1E0000D86C6060F00000003C603C067C00"
9480 DATA 2678
9490 DATA "30307C3030361C000000666666663E0000006666663C18000000C6D6D6FE6C00"
9500 DATA 2246
9510 DATA "0000C66C386CC60000006666663E067C00007E4C18307E000E18187018180E00"
9520 DATA 1802
9530 DATA "18181818181818007018180E1818700076D8000000000000CC33CC33CC33CC33"
9540 DATA 1856
```





Ejemplo del Modo 3.





Ejemplo del Modo 4.

El programa se ha realizado en un IBM pc WGBASIC. Esto significa que cualquier ordenador que disponga de esta versión de BASIC, puede utilizar este programa sin ningún cambio. Para los usuarios del MSX, AMSTRAD y COMMODORE se proponen los siguientes cambios:

COMMODORE

```
8240 REM
8330 GOSUB 9600: LET T=T+R
8550 IF SW<2 THEN LET M$=CHR$ (166): LET W$=" "
8560 IF SW=2 THEN LET M$=" ": LET W$=CHR$ (166)
8710 POKE 214, Y+J-1: POKE 211, X
8730 GOSUB 9600: LET A=R
8930 IF X>40 THEN LET X=1: LET Y=Y+8
```

También hay que incluir el siguiente grupo de líneas.

```
9600 REM
9610 REM ***********************************
9620 REM * PASO DE HEXADECIMAL A DECIMAL *
9630 REM ***********************************
9640 REM
9650 LET R=0
9660 FOR K=1 TO 2
9670 FOR 0=1 TO 16
9680 IF MID $ (Z$, 0, 1) <> MID$ (B$, K, 1) THEN GOTO 7600
9690 LET R=R+INT ((0-1)*(16^(2-K) +0.5: LET 0=16
9700 NEXT 0
9710 NEXT K
9720 RETURN
```

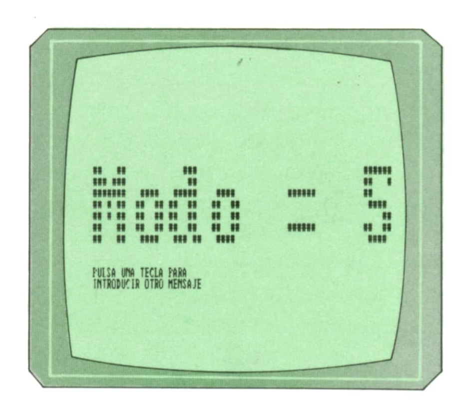
MSX

```
8710 LOCATE X, Y+J-1
8930 IF X>40 THEN LET X=1: LET Y=Y+8
```

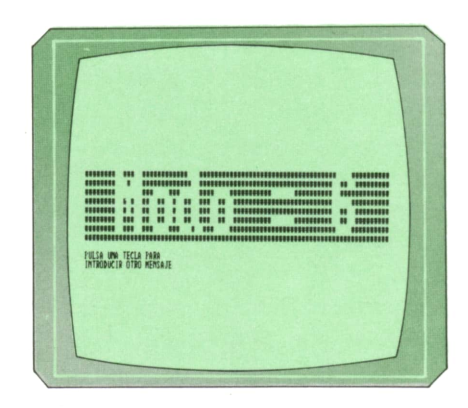
AMSTRAD

8550 IF SW<2 THEN LET M$=CHR$ (143): LET W$=" "
8560 IF SW=2 THEN LET M$=" ": LET W$=CHR$ (143)
8710 LOCATE X, Y+J−1

El programa no es válido para los usuarios del SPECTRUM, para ellos se ha realizado el programa número 2.



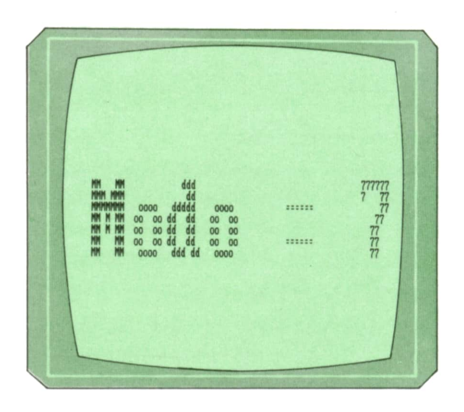

Ejemplo del Modo 5 usando el carácter almohadilla (#).



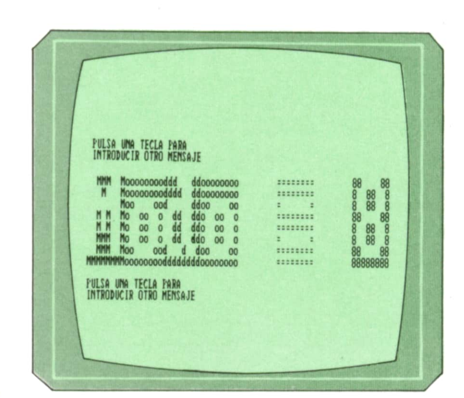

Ejemplo del Modo 6 usando el carácter almohadilla (#).

Para utilizar esta rutina, antes de imprimir ningún mensaje, hay que inicializarla. Esto se consigue haciendo una llamada a la línea 8000. Esta llamada sólo hay que hacerla una vez en el programa. Cuando la rutina retorne, ya está lista para ser utilizada. Los datos que hay que mandarla para que funcione son los siguientes:

A$ = Mensaje a imprimir.
SW = Tipo de impresión.
X = Coordenada en X, donde se empezará a imprimir.
Y = Coordenada en Y, donde se empezará a imprimir.
Si el valor de SW es igual a 5 o a 6, entonces la variable L$ ha de tener almacenado el carácter con el que queremos dibujar.




Ejemplo del Modo 7.




Ejemplo del Modo 8.

A continuación se propone el programa número dos para que se vea cómo funciona y cómo utilizar esta rutina.

```
100 REM ********************************
110 REM * PROGRAMA DEMO PARA EJECUTAR  *
120 REM * LA RUTINA DE LETRAS GIGANTES *
130 REM ********************************
140 REM
150 CLS
160 PRINT "CARGANDO DATAS."
```

```
170 PRINT "ESPERE UN MOMENTO."
180 GOSUB 8000
190 CLS
200 INPUT "MENSAJE = ";A$
210 INPUT "COLUMNA = ";X
220 INPUT "FILA    = ";Y
230 CLS
240 PRINT
250 PRINT "TIPO DE IMPRESION"
260 PRINT "================="
270 PRINT
280 PRINT "1 = IMPRESION NORMAL"
290 PRINT "2 = IMPRESION IMVERTIDA"
300 PRINT "3 = NORMAL DE ASTERISCOS"
310 PRINT "4 = INVERTTIDA DE ASTERISCOS"
320 PRINT "5 = NORMAL DE UN CARACTER A ELEGIR"
330 PRINT "6 = INVERTIDA DE UN CARACTER A ALEGIR"
340 PRINT "7 = NORMAL Y CADA LETRA CON SU CARACTER"
350 PRINT "8 = INVERT. Y CADA LETRA CON SU CARACTER"
360 PRINT
370 PRINT "ELIGE EL TIPO = ";
380 LET B$=INKEY$
390 IF B$="" THEN GOTO 380
400 IF B$<"1" OR B$>"8" THEN GOTO 380
410 PRINT B$
420 PRINT:PRINT
430 LET SW=VAL(B$)
440 IF SW=5 OR SW=6 THEN INPUT "CARACTER= ";L$
450 PRINT
460 PRINT "PULSA UNA TECLA PARA EMPEZAR"
470 LET B$=INKEY$
480 IF B$="" THEN GOTO 470
490 CLS
500 GOSUB 8500
510 PRINT
520 PRINT "PULSA UNA TECLA PARA"
530 PRINT "INTRODUCIR OTRO MENSAJE"
540 LET B$=INKEY$
550 IF B$="" THEN GOTO 540
560 GOTO 190
```

Para que el programa funcione en el COMMODORE hay que realizar los siguientes cambios:

COMMODORE

```
150 PRINT CHR$ (147)
190 PRINT CHR$ (147)
230 PRINT CHR$ (147)
380 GET B$
470 GET B$
490 PRINT CHR$ (147)
540 GET B$
```



Programa ampliador de caracteres para Spectrum

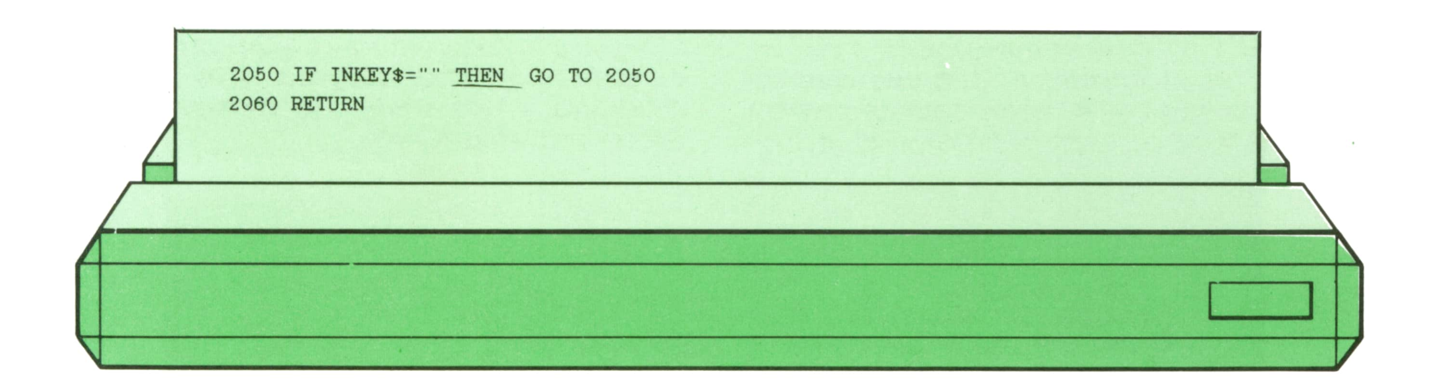
El siguiente programa nos va a permitir tener caracteres ampliados en nuestro SPECTRUM. Estos caracteres podrán ser:

— El doble de ancho.
— El doble de alto.
— El doble de alto y el doble de ancho.

El programa no necesita de muchas explicaciones, ya que todas están inclui-

das en el programa. Lo único que hay que decir es que primero hace falta introducir el programa 4. Una vez que todas las líneas DATA estén correctamente escritas y el programa funcione, grabamos el código máquina que este programa genera a continuación del programa 3. Este programa cargará este código máquina y nos hará una demostración de su funcionamiento.

```
   1 DEF FN p(a,b,c,a$)=USR 50000
  10 BORDER 0: PAPER 0: INK 7: FLASH 0: INVERSE 0: BRIGHT 0
  20 CLEAR 49999: GO SUB 1000
  30 FLASH 1: RANDOMIZE FN p(2,0,4,"AMPLIACION DE CARACTERES")
  40 FLASH 0
  50 RANDOMIZE FN P(0,4,6,"Esto es un")
  60 RANDOMIZE FN p(0,6,6,"ejemplo de")
  70 RANDOMIZE FN p(0,8,5,"utilizacion")
  80 GO SUB 2000
  90 CLS
 100 PRINT "Los caracteres se pueden...."
 110 RANDOMIZE FN p(1,4,8,"ALARGAR")
 120 RANDOMIZE FN P(2,7,10,"ENSANCHAR")
 130 RANDOMIZE FN P(0,10,0,"O LAS DOS COSAS")
 140 GO SUB 2000
 150 CLS
 160 PRINT "Hay tres modos de ampliacion:"
 170 RANDOMIZE FN p(1,4,2,"MODO 1-ALARGAR")
 180 RANDOMIZE FN P(2,7,7,"MODO 2-ENSANCHAR")
 190 RANDOMIZE FN P(0,10,1,"MODO 0-EL DOBLE")
 200 GO SUB 2000
 210 CLS
 220 PRINT "Para utilizarlos debe de usar:"
 230 PRINT : RANDOMIZE FN P(2,3,1,"LET A=FN P(MODO,X,Y,""TEXTO"")")
 240 PRINT ''''"Teniendo una linea que diga:"
 250 RANDOMIZE FN p(2,8,2,"1 DEF FN P(A,B,C,A$)=USR 50000")
 260 PRINT ''''"Y cargando el codigo maquina con"
 270 RANDOMIZE FN p(2,13,2,"LOAD """" CODE 50000,389")
 280 PRINT ''''"Por supuesto,habiendo hecho"
 290 RANDOMIZE FN p(2,18,2,"CLEAR 49999")
 300 PRINT ''''"Antes del LOAD """"CODE "
 310 GO SUB 2000
 320 CLS
 330 PRINT "Los puede poner con los atribu- tos que quiera,mediante  INK y    PA
PER como siempre."
 340 FOR i=1 TO 7
 350 PAPER 8-i: INK i
 360 BEEP .1,I/5: RANDOMIZE FN p(2,11,6,"ESTA CLARO SI O NO ?")
 370 NEXT I
 380 IF INKEY$="" THEN  GO TO 340
 390 BRIGHT 1: INVERSE 1
 400 INK 7: PAPER 0
 410 RANDOMIZE FN P(2,14,5,"MUY BIEN,HASTA LUEGO!!!")
 420 BRIGHT 0: INVERSE 0
 430 FOR G=I TO 100: NEXT G
 440 GO SUB 2000
 999 STOP
1000 REM CARAGA LOS DATA DEL CODIGO MAQUINA
1020 LOAD ""CODE 50000,389
1030 RETURN
2000 REM MENSAJE 'PULSA UNA TECLA'
2020 FLASH 1
2030 RANDOMIZE FN P(1,23,1,"PULSE UNA TECLA")
2040 FLASH 0
```









Presentación del programa de ampliación de caracteres.



Para utilizarlos debe de usar:
LET A=FN P(MODO,X,Y,"TEXTO")
Teniendo una linea que diga:
1 DEF FN P(A,B,C,A$)=USR 50000
Y cargando el codigo maquina con
LOAD "" CODE 50000,389
Por supuesto,habiendo hecho
:CLEAR 49999
Antes del LOAD ""CODE.
PULSE UNA TECLA



```
   1 DEF FN P(A,B,C,A$)=USR 50000
  10 REM ***********************
  20 REM *PROGRAMA CARGADOR DEL*
  30 REM *CODIGO MAQUINA DEL   *
  40 REM *PROGRAMA DE AMPLIA-  *
  50 REM *CION DE CARACTERES   *
  60 REM ***********************
  70 REM
  80 RESTORE 6000
  90 CLS : LET LIN=6000
 100 PAPER 0: INK 6: BORDER 0
 110 CLEAR 49999
 120 PRINT AT 7,9; FLASH 1;"CARGANDO  DATAS"
 130 PRINT AT 10,8; FLASH 1; INVERSE 1;"ESPERA UN MOMENTO"
 140 FOR I=1 TO 392 STEP 8
 150    LET CHEK=0
 160    FOR J=I TO I+7
 170      READ A
 180      LET CHEK=CHEK+A
 190      POKE 49999+J,A
 200    NEXT J
 210    READ TOT
 220    IF TOT<>CHEK THEN  GO TO 1000
 230    LET LIN=LIN+10
 240 NEXT I
 250 CLS
 260 RANDOMIZE FN P(1,1,2,"CODIGO MAQUINA")
 270 RANDOMIZE FN P(1,3,3,"CARGADO EN LA")
 280 RANDOMIZE FN P(1,5,9,"MEMORIA.")
 290 RANDOMIZE FN P(2,10,0,"INTRODUCE UNA CINTA VIRGEN EN EL")
 300 RANDOMIZE FN P(2,12,1,"CASETE Y PONLO  EN POSICION DE")
 310 RANDOMIZE FN P(2,14,3,"GRABACION (RECORD Y PLAY).")
 320 RANDOMIZE FN P(1,19,1,"PULSA UNA TECLA")
 330 RANDOMIZE FN P(1,21,1,"CUANDO LO HAGAS")
 340 PAUSE 0
 350 SAVE "CODIGO"CODE 50000,389
 360 GO TO 9999
1000 CLS
1010 PRINT "ERROR EN LA LINEA ..."
1020  PRINT "                         ";LIN
1030 PRINT
1040 PRINT "REPASA LAS LINEAS DATA"
1050 LIST LIN
6000 DATA 221,42,11,92,221,70,31,221,909
6010 DATA 78,30,221,86,29,221,94,28,787
6020 DATA 221,102,12,221,110,20,58,141,885
6030 DATA 92,8,120,177,32,2,207,2,640
6040 DATA 125,254,32,210,35,196,124,254,1230
6050 DATA 24,210,35,196,205,37,196,221,1124
6060 DATA 126,4,31,56,92,31,56,51,447
6070 DATA 197,213,229,205,9,196,6,8,1063
6080 DATA 205,191,196,205,191,196,213,26,1423
6090 DATA 205,106,196,205,93,196,205,74,1280
6100 DATA 196,205,93,196,205,74,196,209,1374
6110 DATA 19,16,235,225,205,150,196,205,1251
6120 DATA 150,196,209,193,19,11,120,177,1075
6130 DATA 32,206,201,197,213,229,205,9,1292
6140 DATA 196,6,8,205,191,196,26,119,947
6150 DATA 245,205,74,196,241,119,19,205,1304
6160 DATA 74,196,16,242,225,209,19,193,1174
6170 DATA 205,150,196,11,120,177,32,219,1110
6180 DATA 201,197,213,229,205,9,196,6,1256
6190 DATA 8,205,182,196,213,26,205,106,1141
6200 DATA 196,205,93,196,36,209,19,16,970
6210 DATA 243,225,209,193,11,19,205,139,1244
6220 DATA 196,205,139,196,120,177,32,217,1282
6230 DATA 201,205,27,196,229,38,0,111,1007
```

```
6240 DATA 41,41,41,237,91,54,92,25,622
6250 DATA 235,225,201,26,254,32,56,3,1032
6260 DATA 254,128,216,207,10,197,68,77,1157
6270 DATA 120,230,24,246,64,103,120,230,1137
6280 DATA 7,7,7,7,7,7,129,111,282
6290 DATA 229,217,209,122,230,24,203,47,1281
6300 DATA 203,47,203,47,198,88,87,217,1090
6310 DATA 193,201,36,124,230,7,192,62,1045
6320 DATA 32,133,111,216,124,214,8,103,941
6330 DATA 254,88,216,207,4,122,203,63,1157
6340 DATA 178,119,44,123,203,39,179,119,1004
6350 DATA 45,201,197,6,8,17,0,0,474
6360 DATA 23,48,11,203,19,203,18,55,580
6370 DATA 203,19,203,18,24,9,203,19,698
6380 DATA 203,18,167,203,19,203,18,16,847
6390 DATA 231,193,201,44,192,62,8,132,1063
6400 DATA 103,254,88,216,207,4,125,254,1251
6410 DATA 223,32,11,46,0,62,8,132,514
6420 DATA 103,254,88,216,207,4,44,32,948
6430 DATA 4,46,32,24,240,125,230,31,732
6440 DATA 192,125,198,32,111,201,217,8,1084
6450 DATA 18,19,18,19,8,217,201,8,508
6460 DATA 217,18,1,32,0,235,9,119,631
6470 DATA 237,66,35,8,125,230,31,32,764
6480 DATA 1,9,235,217,201,0,0,0,663
```

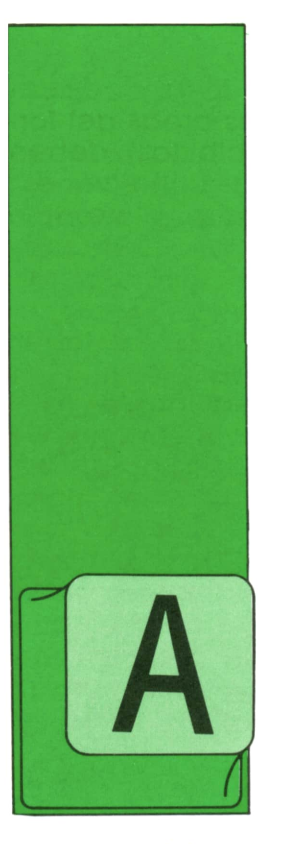




Si el programa número 4 se ha ejecutado bien, nos aparecerá esto en pantalla.

TECNICAS DE ANALISIS

DOCUMENTO DE TOMA DE DATOS



PARTE de la presentación general del documento, ya comentada, es importante diseñar cuidadosamente el cuerpo del impreso para minimizar los riesgos de errores y obtener el máximo interés de las personas que han de cumplimentar el cuestionario.

Hay que seleccionar y preparar cuidadosamente las preguntas, y pensar, cuándo se van a plasmar en un formulario concreto, en las actividades que posteriormente haya que realizar con esa información: codificación, transcripción, etcétera.

Respecto del conjunto de las preguntas, es del máximo interés tener en cuenta varios aspectos:

a) Limitar el número de preguntas, porque un cuestionario excesivamente largo resulta tedioso de rellenar (e incluso puede ser complicado).

b) Incluir todos los datos a obtener; la razón es obvia. Queremos subrayarlo, sin embargo, pues siempre hay que encontrar un punto medio de compromiso con la limitación anterior, para que, obteniendo todos los datos necesarios se intenten condensar las informaciones y no producir un cuestionario excesivamente largo.

c) Incluir preguntas de comprobación. Se consideran tales las que no aportan información adicional, sino que se añaden para contrastar los datos ya incluidos en otras preguntas cuya información es básica en el proceso de que se trate. Si las preguntas fundamentales son claras, se pueden eliminar las preguntas de comprobación, por brevedad; pero en muchos otros casos son de gran utilidad.

d) Ordenar cuidadosamente las preguntas. Hay que procurar que el propio orden de las preguntas ayude a quien ha de responder a dar sus contestaciones. Es importante evitar dar «saltos», en el cuestionario y, por el contrario, abordar los temas sistemáticamente y de tal modo que se planteen primeramente las preguntas genéricas, para descender después a detalles (excepto que las preguntas genéricas sean de «integración»: es decir, que su respuesta se «deduzca» de los datos parciales aportados en las preguntas de detalle, en cuyo caso el orden debe ser el inverso).

e) Incluir preguntas de «descanso». Estas se refieren a temas menores o de respuesta más evidente: con eso, la persona que responde el cuestionario relaja la tensión a que le pueda someter una «batería» de preguntas importantes seguidas. Hay que tener en cuenta en este caso, también, el comentario hecho anteriormente sobre la brevedad del formulario.

Por otro lado, hay que estudiar cuidadosamente cómo se plantean las preguntas. Se pueden diseñar preguntas abiertas (de contestación libre), preguntas tipo «menú» (se sugieren numerosas respuestas, aunque se suele dejar siempre una última opción para respuestas adicionales libres) o preguntas de tipo

cerrado (hay que elegir una respuesta entre las que se ofrecen). Conviene que las preguntas se diseñen, siempre que sea posible, de tipo cerrado o, a lo sumo, de «menú». Incluso, si se puede, es útil tender hacia preguntas que requieran una contestación binaria (sí, no).

Sin embargo, conviene en todo momento que primen sobre los comentarios anteriores la consideración de las condiciones básicas que deben cumplir las preguntas: que sean claras y no ambiguas, que sean breves y concisas, que sean fácilmente entendibles, etc.

Respecto del espacio reservado para las respuestas y para la codificación de los datos (si ha lugar) deben tenerse en cuenta algunas consideraciones adicionales:

a) La zona de las respuestas debe ser suficientemente amplia para contener cuanto necesite poner la persona que debe responder el cuestionario, aunque suele ser útil limitar la longitud de las respuestas, en la medida de lo posible.

b) Hay que tener cuidado para que el espacio reservado a la respuesta no induzca a error (o duda) a quien debe contestar. Por ejemplo, una zona donde debe ir un teléfono debe tener previsto espacio para el prefijo provincial (aunque luego, de hecho, la mayoría de respuestas no lo incluyan); una zona donde se debe responder una cifra debe tener una longitud suficiente para el mayor número aceptable como respuesta, etc.

c) Aunque a veces es útil marcar las posiciones de cada letra o número (mediante cuadros o pequeñas rayas verticales), hay que procurar no constreñir las respuestas por este motivo.

d) La zona reservada a las respuestas debe aparecer inmediata a la pregunta, para que quien responda el cuestionario no dude dónde debe contestar.

e) Si alguna pregunta va a ser codificada posteriormente o si se va a incluir después algún código o dato adicional a partir de las respuestas dadas, la zona prevista para estos códigos o informaciones adicionales debe estar claramente marcada (quizá otro color o un sombreado sea la solución) para que nunca sea utilizada por quien contesta el cuestionario.

Por otro lado, respecto de las codificaciones establecidas (y las áreas del formulario destinadas a recibirlas) deben tenerse en cuenta algunas condiciones:

a) Hay que procurar que el espacio destinado a las codificaciones esté contiguo al área donde se incluirán las respuestas.

b) Siempre que se pueda, se deben incluir los valores de los códigos junto al espacio reservado a la codificación (aun teniendo en cuenta los comentarios ya hechos sobre la inclusión de notas, instrucciones, etc., en el cuerpo del impreso).

c) Cuando sea posible (especialmente en las preguntas tipo «menú» o cerradas y en las de respuesta binaria) conviene pedir directamente a quien conteste el cuestionario que codifique las respuestas, mediante las clásicas expresiones «táchese la que no proceda», «márquese con una cruz la respuesta correcta», etc., pues, aparte de evitar una manipulación posterior del documento, elimina causas de error.

Por último, conviene comentar la necesidad de tener en cuenta, cuando se diseñe un documento de toma de datos, cómo se han de transcribir los datos posteriormente y, para ello:

a) Prever que los datos sean transcritos en el orden en que aparecen en el cuestionario.

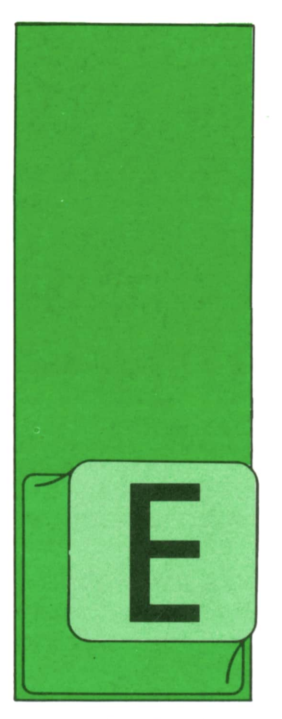
b) Hacer el diseño de tal modo que el transcriptor no deba «andar recorriendo» con la vista el documento, sino que los datos aparezcan relativamente agrupados.

c) Que se incluyan indicaciones para transcripción (número de columnas de registro, comienzo o final de registro, marcas adicionales de identificación, etcétera.

En general, hay que procurar que todas las tareas que comporta la toma de datos (contestación del cuestionario, codificación, transcripción, etc.) sean lo más automáticas posible para minimizar la comisión de errores.

TECNICAS DE PROGRAMACION

INSTRUCCIONES CONDICIONALES

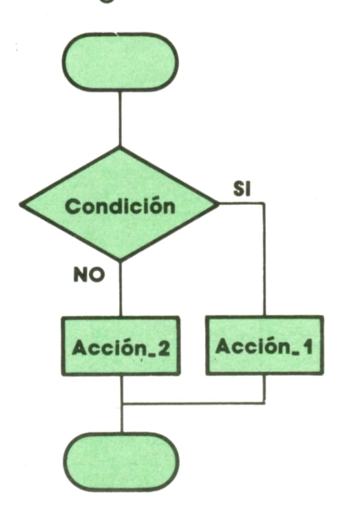


L segundo elemento fundamental de la programación «bien estructurada» es la instrucción condicional, de la que existen varias formas posibles. La más general es la siguiente:

SI condición
  ENTONCES acción-1
  EN-CASO-CONTRARIO acción-2

es decir: «si se cumple cierta condición, se ejecuta acción-1. Pero si no se cumple, se ejecuta acción-2».

El organigrama de esta instrucción condicional es el siguiente:



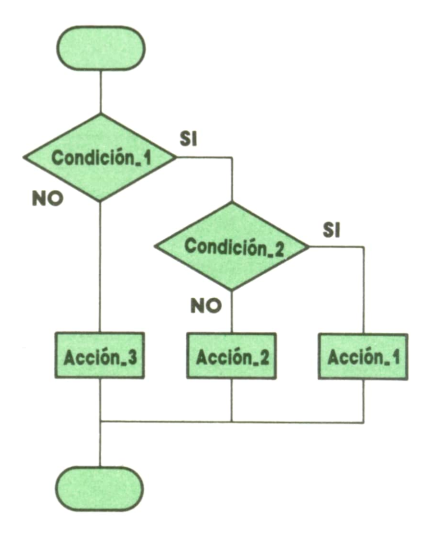
Normalmente, la condición que aparece en una instrucción condicional puede ser cualquier expresión de resultado lógico, como las que vimos en el capítulo 12, que pertenecían a dos grupos principales: comparaciones entre expresiones de cualquier tipo, o unión mediante funciones lógicas de dos o más expresiones lógicas cualesquiera.

En cambio, los elementos ejecutables (acción-1, acción-2) representan instrucciones de cualquier clase: asignaciones, instrucciones condicionales, bucles, bloques secuenciales, llamadas a subrutinas, instrucciones de transferencia, órdenes de entrada o de salida... Una instrucción que pueda efectuarse de modo independiente en otro lugar del programa puede ser siempre parte de una instrucción condicional, ya sea como acción-1 o como acción-2.

En particular, cualquiera de las dos instrucciones ejecutables que forman parte de una instrucción condicional (o las dos al mismo tiempo) puede ser una instrucción condicional. Esto significa que las instrucciones condicionales pueden encadenarse para formar estructuras más complicadas. Veamos algunos ejemplos:

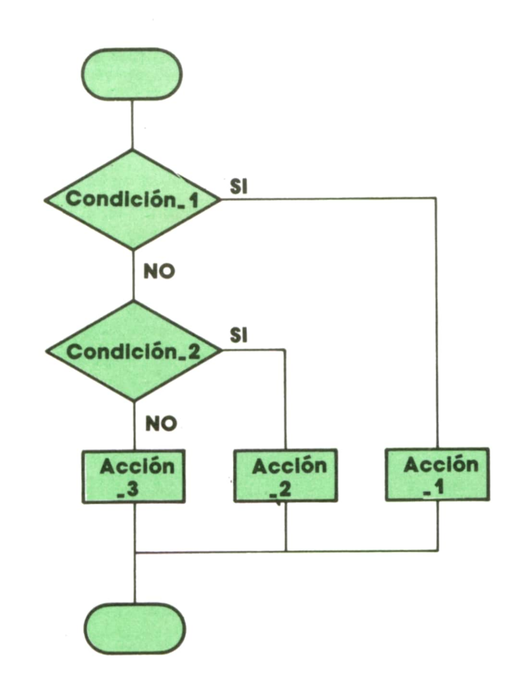
SI condición-1
  ENTONCES SI condición-2
    ENTONCES acción-1
    EN-CASO-CONTRARIO acción-2
  EN-CASO-CONTRARIO acción-3

cuyo organigrama es:



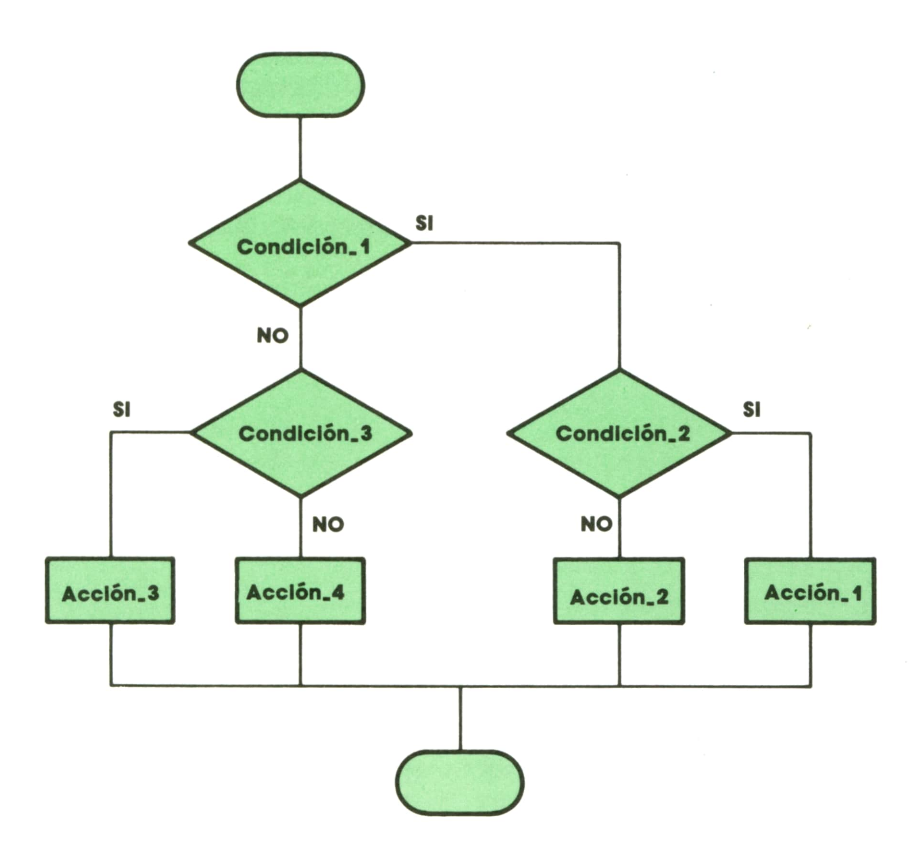
SI condición-1
ENTONCES acción-1
EN-CASO-CONTRARIO
SI condición-2
ENTONCES acción-2
EN-CASO-CONTRARIO acción-3

cuyo organigrama es:



SI condición-1
ENTONCES SI condición-2
ENTONCES acción-1
EN-CASO-CONTRARIO acción-2
EN-CASO-CONTRARIO
SI condición-3
ENTONCES acción-3
EN-CASO-CONTRARIO acción-4

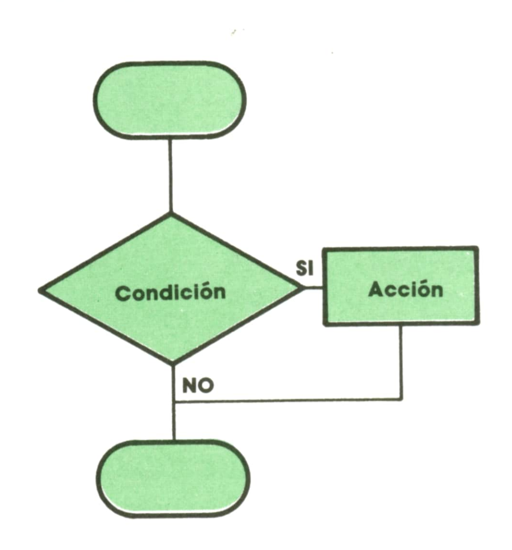
cuyo organigrama es:



La instrucción condicional podría complicarse aún más, introduciendo nuevos términos condicionales en lugar de cualquiera de las acciones anteriores. También puede simplificarse, prescindiendo de la parte EN-CASO-CONTRARIO, con lo que la instrucción quedaría reducida a:

SI condición
  ENTONCES acción

cuyo organigrama es:



y que puede describirse así: «si se cumple la condición, ejecutar la acción, pero no hacer nada si la condición no se cumple».

También pueden mezclarse las instrucciones condicionales reducidas con las completas, como en el siguiente ejemplo:

SI CONDICION-1
  ENTONCES acción-1
  EN-CASO-CONTRARIO
    SI condición-2
    ENTONCES acción-2

cuyo organigrama dejamos como ejercicio al lector.

Consideremos también el siguiente ejemplo:

SI condición-1
  ENTONCES SI condición-2
    ENTONCES acción-1
    EN-CASO-CONTRARIO acción-2

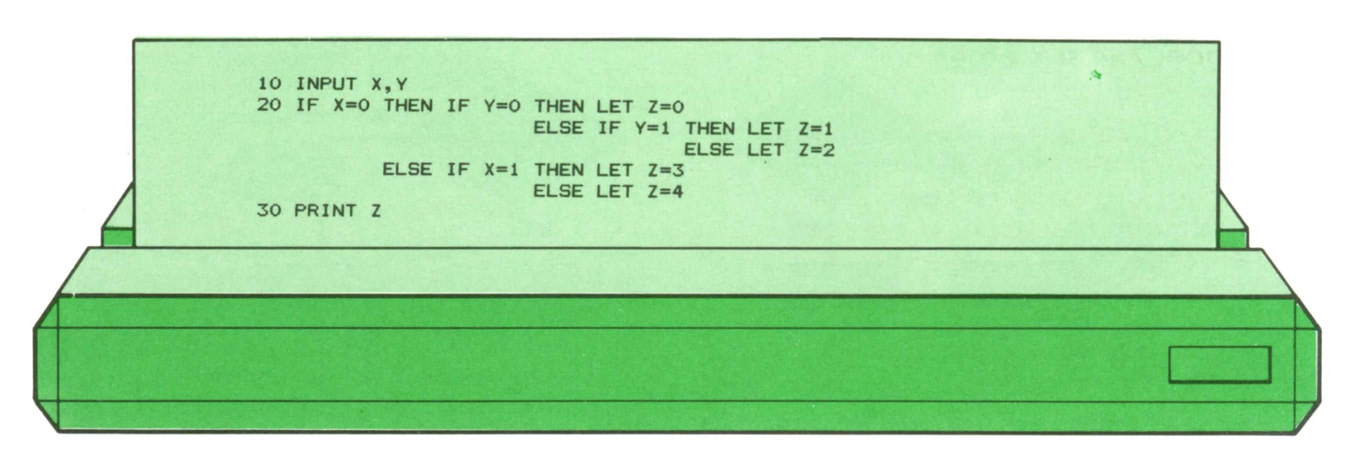
Tenemos aquí dos instrucciones condicionales, una de las cuales está incompleta. Pero, ¿a cuál de ellas pertenece la cláusula EN-CASO-CONTRARIO? Podría ser a la primera, pero también podría ser a la segunda. En general, esto depende del lenguaje de programación de que se trate. Volveremos sobre ello más adelante.

Veamos ahora cómo se programan las instrucciones condicionales en los tres lenguajes que estamos utilizando como elemento de comparación: BASIC, PASCAL y APL.

En BASIC se utilizan las palabras reservadas inglesas IF, THEN y ELSE para representar lo que hasta ahora hemos llamado SI, ENTONCES y EN-CASO-CONTRARIO. Por tanto, la forma más general de la instrucción condicional será:

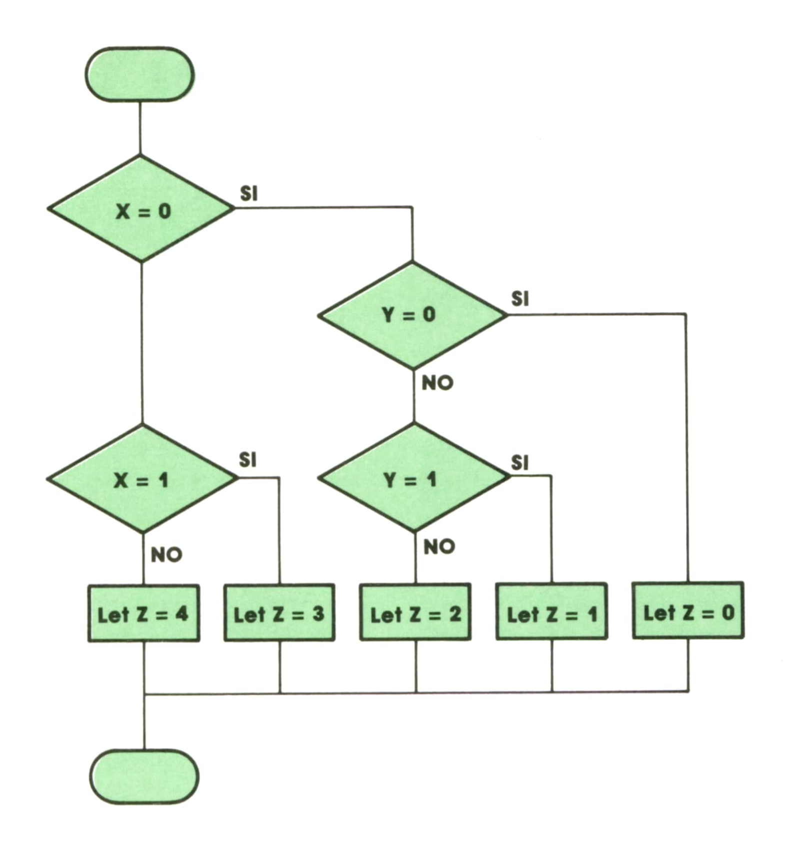
IF condición
  THEN acción-1
  ELSE acción-2

Veamos un ejemplo de un programa BASIC que utiliza una instrucción condicional relativamente compleja:



```
10 INPUT X,Y
20 IF X=0 THEN IF Y=0 THEN LET Z=0
                      ELSE IF Y=1 THEN LET Z=1
                                  ELSE LET Z=2
          ELSE IF X=1 THEN LET Z=3
                      ELSE LET Z=4
30 PRINT Z
```

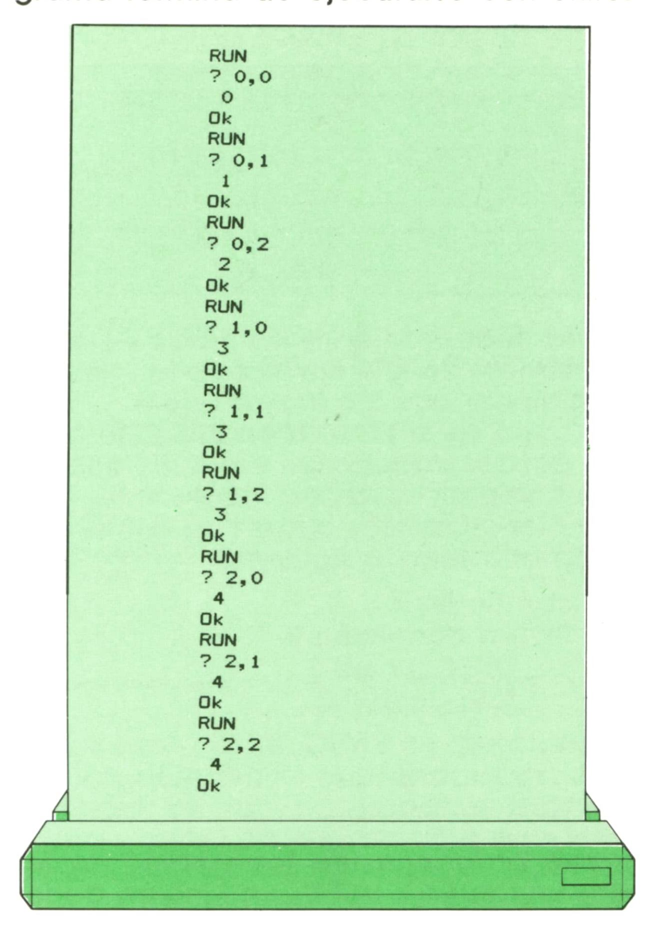
Este programa lee los valores de las dos variables X e Y y, en función de los mismos, calcula el valor de Z en la instrucción 20, cuyo organigrama es:



Obsérvese que el organigrama corresponde a una sola instrucción BASIC. En el programa hemos descompuesto la instrucción en cinco líneas diferentes, con objeto de que apareciera clara la estructura de la instrucción y la relación de las diferentes partes THEN y ELSE entre sí. En definitiva, la instrucción 20 realiza la siguiente función:

— Hacer Z=0 si X e Y son iguales a 0.
— Hacer Z=1 si X es 0, pero Y es 1.
— Hacer Z=2 si X es 0, pero Y no es ni 0 ni 1.
— Hacer Z=3 si X es 1, cualquiera que sea Y.
— Hacer Z=4 si X no es ni 0 ni 1, cualquiera que sea Y.

Vamos a ver que, efectivamente, el programa funciona así, ejecutándolo varias veces para distintos valores de X e Y. Cada vez que aparece la palabra RUN, comienza de nuevo a ejecutarse el programa. Las líneas que comienzan por una interrogación contienen nuestras respuestas a la petición de los datos X e Y (en este orden, separados por una coma). La línea siguiente contiene el valor que la instrucción 20 ha calculado para Z, que ha sido escrito en la pantalla por la instrucción 30 del programa. La palabra Ok aparece cada vez que el programa termina de ejecutarse con éxito.



En BASIC, la instrucción condicional puede ser incompleta:

IF condición
  THEN acción-1

De hecho, la forma más frecuente que adopta este tipo de instrucción en los programas BASIC es aquélla en que «acción-1» se sustituye por una instrucción de transferencia:

IF condición THEN GOTO nnnn

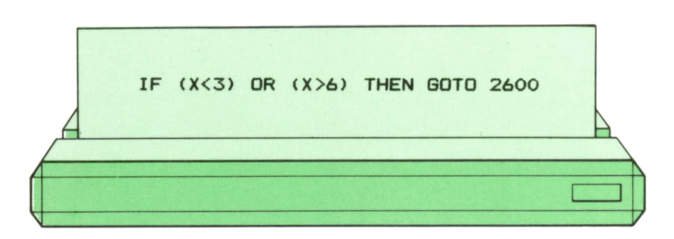
donde nnnn es un número de línea. El efecto de esta instrucción es provocar un salto de la ejecución a la línea número nnnn si se cumple la condición indicada, mientras que la ejecución continuará con la instrucción siguiente a la condicional si la condición no se cumple. En algunos intérpretes o compiladores BASIC, la instrucción anterior puede simplificarse así:

IF condición GOTO nnnn

o también de la siguiente manera:

IF condición THEN nnnn

Veamos ahora un ejemplo de una instrucción condicional cuya condición es relativamente compleja:

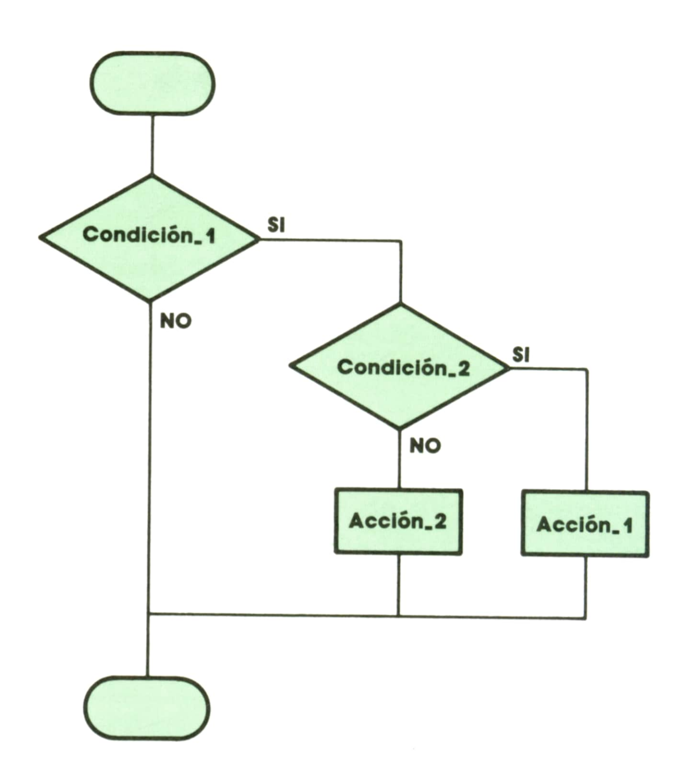


Esta línea transfiere control a la línea de número 2600 si el valor de la variable X es menor que 3 o mayor que 6.

¿Cómo se interpreta en BASIC la mezcla de dos instrucciones condicionales, una completa y la otra incompleta, que mencionamos más arriba? La forma que adoptaría sería la siguiente:

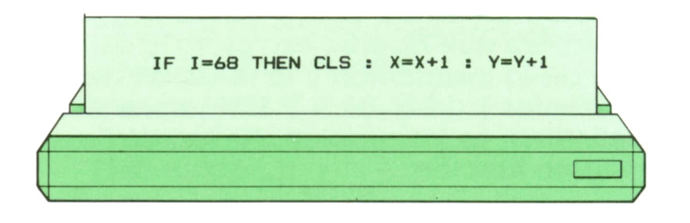
IF condición-1
  THEN IF condición-2
    THEN acción-1
    ELSE acción-2

Pues bien: en BASIC, en un caso como éste, se supone que la cláusula ELSE va siempre emparejada con la cláusula THEN más próxima que no tenga pareja. Por tanto, el organigrama correspondiente a ese ejemplo sería el siguiente:

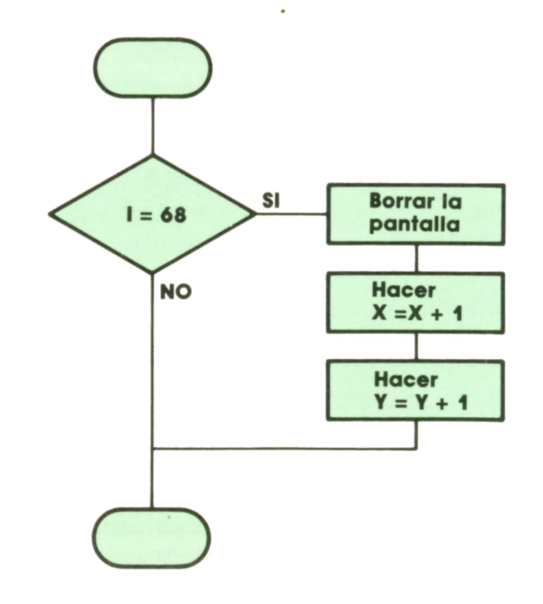


y es el primer THEN de la instrucción el que se queda sin pareja.

Por último, veamos cómo se puede introducir un bloque secuencial en una instrucción condicional BASIC:

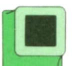


cuyo organigrama es:

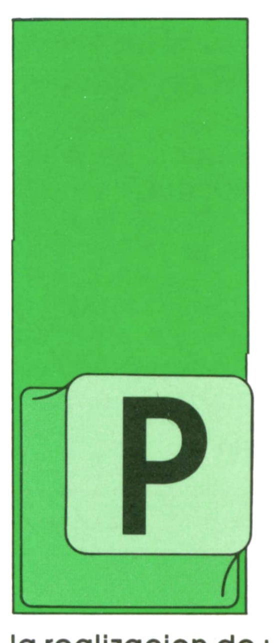


APLICACIONES

MULTIPLAN: EJEMPLO



Inventario



PRESENTAMOS en este capítulo un ejemplo sencillo para facilitar la comprensión de las hojas de cálculo y en concreto del Multiplan. Nuestro ejemplo se va a basar en la realizacion de una hoja de activo y pasivo, lo que se suele conocer con el nombre de inventario.

Se trata de un formulario muy frecuente, cuya configuración básica puede encontrarse en cualquier libro de contabilidad. Hay que ir rellenando la columna de parciales y calculando cada uno de los totales hasta obtener el total activo y el pasivo. Restando éstos se obtendrá el capital líquido.

Para transferirlo a la hoja de cálculo, vamos a procurar mantener su forma, ya que así facilitaremos la tarea de cumplimentar los datos a los usuarios. Esto, que puede parecer carente de importancia, es una de las premisas que deben seguirse para que el trabajo realizado en el ordenador no produzca errores. Si pensamos en el cambio de mentalidad que contables y empresarios han tenido que sufrir para adaptarse a la informática, debemos comprender la importancia de conseguir que ésta se asemeje a las fórmulas preestablecidas en sus campos de trabajo; de aquí la importancia de conseguir que la hoja de cálculo que haya en activo y pasivo tenga la estructura del formulario correspondiente que el contable ha rellenado toda la vida.

Comenzamos por crear con literales (comando Alfa) el esquema de cómo va a ser nuestra hoja de trabajo.

```
ST        INVENTARIO

INVENTARIO FINAL

ACTIVO                              IMPORTES
                              PARCIALES  TOTALES
----------------------------------------------------
CAJA..........................
  Existencias en efectivo

MERCADERIAS...................
  Valor existencias almacén:
Material A....................
Material B....................
Material C....................

MOBILIARIO....................
Valor actual con reducción
10 por 100 por amortización

EFECTOS A COBRAR..............
Valor nominal de una L/. a 15
días fecha

DEUDORES VARIOS...............
Juan Martínez, de Bilbao
Francisco Pérez, de Madrid

------------------------------

     TOTAL ACTIVO.............

------------------------------


PASIVO                              IMPORTES
                              PARCIALES TOTALES
----------------------------------------------------
ACREEDORES VARIOS.............
Pedro García, de Sevilla
Lucindo Ríos
Angela Chaning

EFECTOS A PAGAR...............
Valor nominal de un L/. a 60
días fecha

CAPITAL.......................
Por el capital inicial

PERDIDAS Y GANANCIAS..........
Beneficio en el presente

------------------------------

     TOTAL PASIVO.............

------------------------------

RESUMEN

     TOTAL ACTIVO.............
     TOTAL PASIVO.............
```

Observamos que la longitud de algunas celdas ha tenido que ser cambiada para que los literales se vean completamente. Por otra parte hay que reseñar que las celdas que son iguales se obtienen por copia de otras, ya que no es necesario volver a realizarlas.

Una vez terminada esta fase se pasa al perfeccionamiento de la hoja:

— Separación de las columnas parciales y totales, generando columnas (comando Genera) que se reducen a tres caracteres de longitud (comando Format/Longitud) e introduciendo las líneas correspondientes a la separación (comando Alfa).

— Para crear la línea vertical se puede utilizar el comando Replicar/Vertical que copia tantas veces como se le indique el contenido de la celda señalada.

— Asimismo hay que variar la longitud de algunas celdas para que los literales se vean completamente. Las columnas anchas tienen una longitud de 30 caracteres.

ST INVENTARIO

INVENTARIO FINAL

ACTIVO	IMPORTES PARCIALES	IMPORTES TOTALES
CAJA........................		
Existencias en efectivo		
MERCADERIAS...................		
Valor existencias almacén:		
Material A....................		
Material B....................		
Material C....................		
MOBILIARIO.....................		
Valor actual con reducción 10 por 100 por amortización		
EFECTOS A COBRAR.............		
Valor nominal de una L/. a 15 días fecha		
DEUDORES VARIOS...............		
Juan Martínez, de Bilbao		
Francisco Pérez, de Madrid		
TOTAL ACTIVO............		---------

PASIVO	IMPORTES PARCIALES	IMPORTES TOTALES
ACREEDORES VARIOS............		
Pedro García, de Sevilla		
Lucindo Ríos		
Angela Chaning		
EFECTOS A PAGAR..............		
Valor nominal de un L/. a 60 días fecha		
CAPITAL......................		
Por el capital inicial		
PERDIDAS Y GANANCIAS.........		
Beneficio en el presente		
TOTAL PASIVO............		---------
RESUMEN		
TOTAL ACTIVO............		
TOTAL PASIVO............		
CAPITAL LIQUIDO.........		---------

Evidentemente estamos logrando que el formulario tenga aspecto muy semejante al real.

Una vez preparado el esquema podemos pasar a la introducción de fórmulas. Para ello utilizamos el comando Edit. Observará que el simple movimiento del cursor hace aparecer en la fórmula el nombre de la celda.

Las fórmulas de este ejemplo están compuestas por fórmulas con operaciones aritméticas sencillas (+ y -), y la función suma (:), que permite sumar un rango de celdas.

En principio, como no se han introducido datos, aparece en la celda en la que se introduce la fórmula un cero. Si queremos ver en la pantalla las fórmulas, podemos utilizar el comando format para visualizar en pantalla las fórmulas, adquirirá la siguiente forma:

```
#1  2              3             4         5               6        7
18 " ||"                         " |" 3600                 " ||"
19 " ||"                         " |"                      " ||"
20 " ||"                         " |"                      " ||"
21 " ||"                         " |"                      " ||"
22 " ||"                         " |" 6700                 " ||"
23 " ||"                         " |"                      " ||"
24 " ||"                         " |"                      " ||"
25 " ||"                         " |"                      " ||"
26 " ||"                         " |" SUMA(R[+1]C[-2]:R[+2" ||"
27 " ||" 20000                   " |"                      " ||"
28 " ||" 32000                   " |"                      " ||"
29 " ||"                         " |"                      " ||"
30 " ||"                         " |"                      " ||"
31 " ||"                         " |"                      " ||"
32 " ||"                         " |" SUMA(R[-23]C:R[-6]C)" ||"
33 " ||"                         " |" "          "         " ||"
34 " ||"                         " |"                      " ||"
35
MANDATO: Alfa Blan Clasif Direc Edit Format Genera HoAyu Imprimir Limit
         Mov Nomb Opcion Proteg Quitar Replic Salir Transf Valor Xterno
Seleccionar opción o pulsar inicial de mandato
R34C3                                 98% Libre ML  Multiplan: B:INVERTA
```

Podemos pasar a la comprobación del funcionamiento con datos conocidos o archivos históricos.

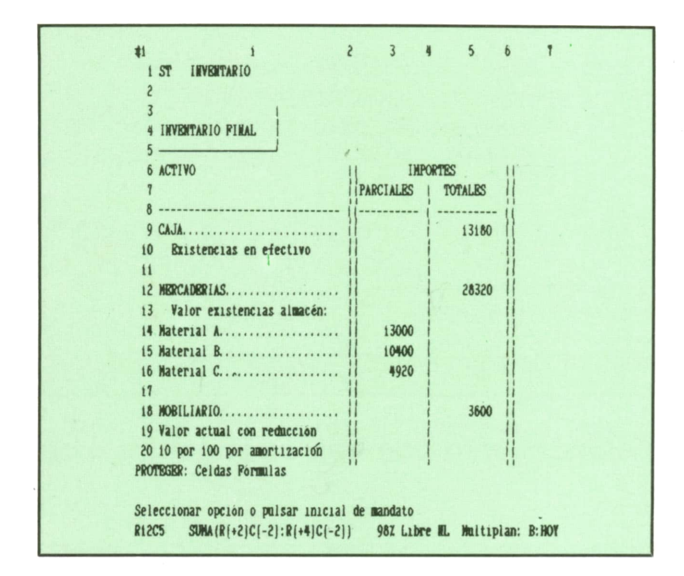
ST INVENTARIO

INVENTARIO FINAL

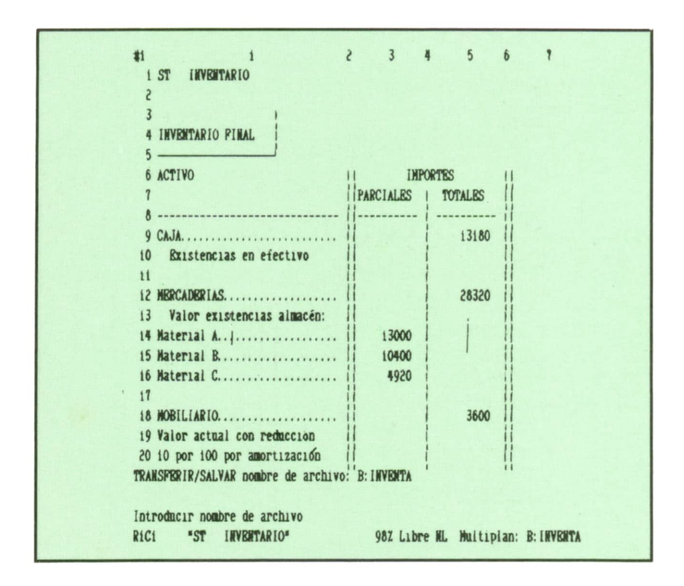
ACTIVO	IMPORTES PARCIALES	IMPORTES TOTALES
CAJA..........................		13180
Existencias en efectivo		
MERCADERIAS...................		28320
Valor existencias almacén:		
Material A....................	13000	
Material B....................	10400	
Material C....................	4920	
MOBILIARIO....................		3600
Valor actual con reducción 10 por 100 por amortización		
EFECTOS A COBRAR..............		6700
Valor nominal de una L/. a 15 días fecha		
DEUDORES VARIOS...............		52000
Juan Martínez, de Bilbao	20000	
Francisco Pérez, de Madrid	32000	
TOTAL ACTIVO............		103800

PASIVO	IMPORTES PARCIALES	IMPORTES TOTALES
ACREEDORES VARIOS.............		14500
Pedro García, de Sevilla	2700	
Lucindo Ríos	1800	
Angela Chaning	10000	
EFECTOS A PAGAR...............		5000
Valor nominal de un L/. a 60 días fecha		
CAPITAL.......................		43500
Por el capital inicial		
PERDIDAS Y GANANCIAS..........		4270
Beneficio en el presente		
TOTAL PASIVO............		67270
RESUMEN		
TOTAL ACTIVO............		103800
TOTAL PASIVO............		67270
CAPITAL LIQUIDO.........		36530

Una vez asegurados del funcionamiento podemos poner la hoja en manos del usuario, no sin antes cumplimentar la adecuada documentación de generación y uso, así como cerciorarse de la protección de las celdas cuyo contenido es una fórmula.

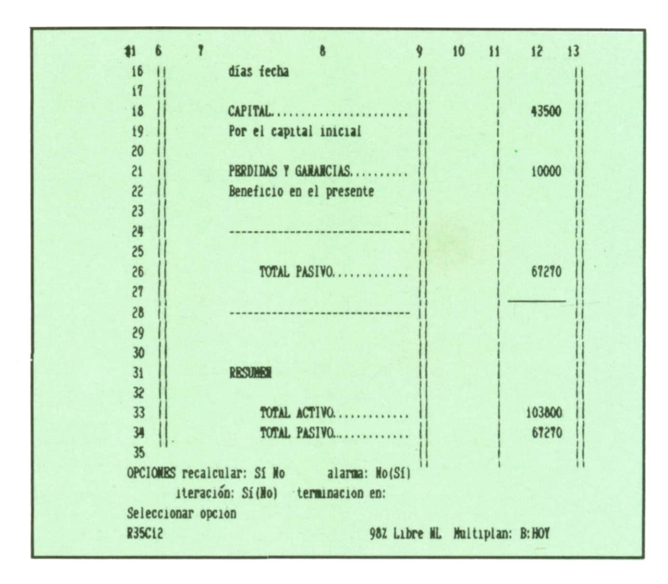


Realizamos una grabación del modelo:



Ya se puede utilizar la hoja, por ejemplo, se cambian algunos de los valores introducidos previamente. Si el comando Recalcular no se encuentra activo, hacemos referencia a él para que todos los valores y resultados se actualicen.

Una vez mostrado el ejemplo, pasamos a detallar un punto importante: la impresión de la hoja.



Si observamos la longitud en caracteres excede en mucho la longitud habitual de una impresora. Para remediar esto y poder imprimir todo el modelo tenemos que recurrir a las siguientes posibilidades que en general son aplicables a los restos de las hojas de cálculo:

Se aconseja, en primer lugar, utilizar el modo de impresión comprimida, éste puede obtenerse de las siguientes formas:

— Mediante un programa adicional que se carga al principio de la sesión de trabajo y actúa como programa residente en memoria, permitiendo alteraciones en la impresión: modo comprimido, tipo de letra, doble impresión, líneas por página, caracteres por pulgada, etc.

— Mediante opciones de la misma hoja (no existen en Multiplan) que permiten el modo comprimido.

— Mediante el envio de los caracteres de control de la impresora correspondiente, varían de una impresora a otra, pero son típicos los siguientes:

Alt 15 para letra comprimida.
Alt 14 para letra expandida.
Alt 12 para restaurar.

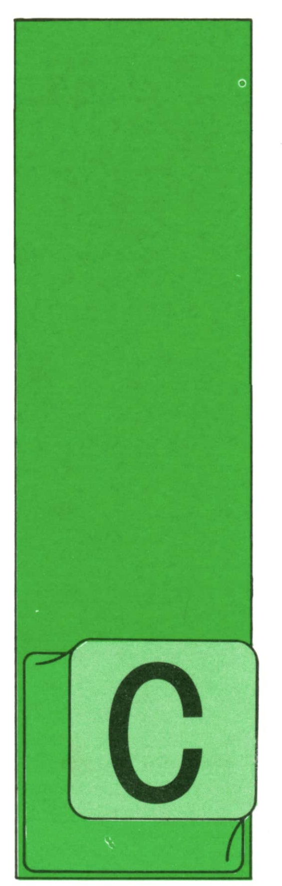
Como paso adicional en el Multiplan tenemos que indicar el número de caracteres que deseamos que tenga la hoja. El comando correspondiente es Imprimir/Márgenes. En el ejemplo para el modelo final se ha utilizado una longitud de 130 caracteres, dejando en el margen izquierdo sólo dos caracteres.

Por último, mostraremos el listado de la hoja de trabajo del ejemplo:

```
 1 "ST   INVENTARIO"
 2
 3 "                        |"
 4 "INVENTARIO FINAL        |"
 5 "________________________|"
 6 "ACTIVO"
 7
 8 "-------------------------------"
 9 "CAJA..........................."
10 "  Existencias en efectivo"
11
12 "MERCADERIAS...................."
13 "  Valor existencias almacén:"
14 "Material A...................."
15 "Material B...................."
16 "Material C...................."
17
18 "MOBILIARIO...................."
19 "Valor actual con reducción"
20 "10 por 100 por amortización"
21
22 "EFECTOS A COBRAR.............."
23 "Valor nominal de una L/. a 15"
24 "días fecha"
25
26 "DEUDORES VARIOS..............."
27 "Juan Martínez,de Bilbao"
28 "Francisco Pérez,de Madrid"
29
30 "-------------------------------"
31
32 "      TOTAL ACTIVO............."
33
34 "-------------------------------"
        2           3           4           5           6
 1
 2
 3
 4
 5
 6 " ||" "         IM"     "POR" "TES"                 " ||"
 7 " ||" "PARCIALES"       " |"  "TOTALES"             " ||"
 8 " ||" "-----------"     " |"  "-----------"         " ||"
 9 " ||"                   " |"  13180                 " ||"
10 " ||"                   " |"                        " ||"
11 " ||"                   " |"                        " ||"
12 " ||"                   " |"  SUMA(R[+2]C[-2]:R[+   " ||"
                                 4]C[-2])
13 " ||"                   " |"                        " ||"
14 " ||" 13000             " |"                        " ||"
15 " ||" 10400             " |"                        " ||"
16 " ||" 4920              " |"                        " ||"
17 " ||"                   " |"                        " ||"
18 " ||"                   " |"  3600                  " ||"
19 " ||"                   " |"                        " ||"
20 " ||"                   " |"                        " ||"
21 " ||"                   " |"                        " ||"
22 " ||"                   " |"  6700                  " ||"
23 " ||"                   " |"                        " ||"
24 " ||"                   " |"                        " ||"
25 " ||"                   " |"                        " ||"
26 " ||"                   " |"  SUMA(R[+1]C[-2]:R[+   " ||"
                                 2]C[-2])
27 " ||" 20000             " |"                        " ||"
28 " ||" 32000             " |"                        " ||"
29 " ||"                   " |"                        " ||"
30 " ||"                   " |"                        " ||"
31 " ||"                   " |"                        " ||"
32 " ||"                   " |"  SUMA(R[-23]C:R[-6]C   " ||"
                                 )
33 " ||"                   " |"  "  -----------"       " ||"
34 " ||"                   " |"                        " ||"
```

PASCAL

PROCEDIMIENTOS RECURSIVOS



UANDO estudiamos en qué casos un procedimiento o función podía llamar a otro, vimos que era posible que se llamaran a sí mismos; cuando un procedimiento o función utiliza esta posibilidad, se dice que es «recursivo». Vamos a ver cómo funciona este tipo de subprograma por medio del ejemplo que casi siempre se utiliza para ello:

Supongamos que hubiera que programar la función matemática «factorial». El factorial de un número entero N se escribe como N! y es igual al producto de N por N-1 por N-2... hasta llegar a 1. Por ejemplo:

1! es 1.
2! es 2 por 1, igual a 2.
3! es 3 por 2 por 1, igual a 6.
4! es 4 por 3 por 2 por 1, igual a 24.

Como se puede ver, el factorial de un número cualquiera es igual a él mismo multiplicado por el factorial del número anterior:

2! es igual a 2 por 1!
3! es igual a 3 por 2!
4! es igual a 4 por 3!
5! es igual a 5 por 4!

Por ello, el programa para calcular el factorial de, digamos, 10, sería:

1 — Calcular el factorial de 9.
2 — Multiplicarlo por 10.

a su vez, el punto 1 (calcular el factorial de 9) sería:

A — Calcular el factorial de 8.
B — Multiplicarlo por 9.

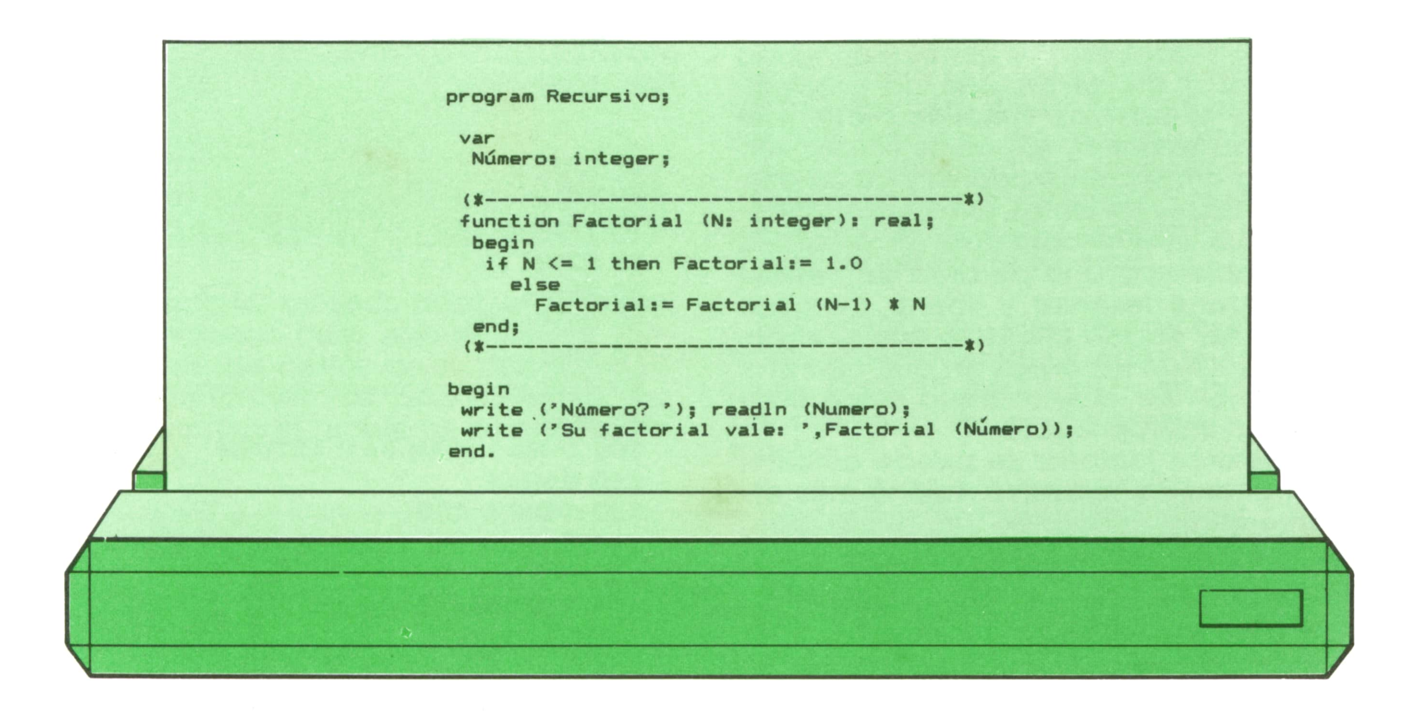
y así hasta llegar a «calcular el factorial de 1» que consistiría simplemente en tomar un 1. Un programa PASCAL con esta estructura debería tener una función para calcular el factorial de 10, otra para el de 9, etc.; evidentemente es una forma poco práctica de programar (imaginemos cómo sería para el factorial de 20, 30...). Sería más interesante tener una función única a la que se pasara como parámetro el número cuyo factorial deseamos; el programa para calcularlo sería algo así:

«Calcular el factorial del parámetro»:
— Si el parámetro vale 1 devolver 1 y si no:
1 — Calcular el factorial del parámetro menos 1.
2 — Multiplicarlo por el propio parámetro y devolverlo.

y para ejecutar el punto 1 se utilizaría exactamente el mismo programa, pero pasándole como parámetro el original menos 1.

Esto es lo que se denomina un programa recursivo, pues en la descripción de sus pasos se hace referencia (o se utiliza) a sí mismo.

El PASCAL permite la escritura de procedimientos y funciones recursivos sin más, por las buenas:



Se ha utilizado el tipo real para poder obtener resultados mayores de los posibles con el tipo integer (el factorial de cero vale 1 y por ello se ha utilizado la comparación «menor o igual»).

Vamos a explicar de manera resumida cómo es posible que una función (o procedimiento) se llame a sí misma:

Imaginemos que se ejecuta Factorial (3). En el momento de llamar a la función se toma una porción de memoria para la variable local N en la que se guarda un 3.

Posteriormente, y ya dentro de la función, como N es mayor que 1 se ejecuta Factorial (N-1); por ello se toma una nueva porción de memoria para N y en ella se guarda un 2 (como todavía no se ha regresado al programa principal, la porción que se tomó para guardar el 3 sigue ocupada).

Al ejecutarse Factorial (2), cada vez que aparece N, el PASCAL busca la más reciente porción de memoria con ese nombre, que es la que alberga un 2. Allí, al ser N otra vez mayor que 1, se ejecuta Factorial (N-1) tomándose una nueva porción de memoria para N, en la que se guarda un 1.

Con Factorial (1), y debido a la instrucción IF, se devuelve 1.0 y se acaba su ejecución. Entonces se libera la memoria en que se guardó el 1 y se vuelve al punto desde el que se llamó, es decir, en medio de la ejecucción de Factorial (2).

Allí, el resultado de Factorial (1) se multiplica por el valor de N que, al haberse liberado la memoria que guardaba el 1, es de nuevo 2. Tras esto se acaba la ejecución de Factorial (2) recuperándose la memoria en que se guardó el 2 y devolviéndose 1.0 * 2 al punto medio de Factorial (3).

Entonces, 2.0 se multiplica por 3 (pues ahora la más reciente porción de memoria con el nombre N es la que contiene un 3) y se devuelve 6.0 como resultado de Factorial (3), quedando libre la memoria en que se guardó el 3.

Por supuesto, la posibilidad de escribir procedimientos recursivos se debe en gran parte a la existencia de variables locales (es decir, aquéllas que se crean en el momento de llamarse a un procedimiento y cuya memoria queda libre tras ejecutarse éste).

La necesidad de memoria puede llegar a ser grande, pues la cantidad utilizada para variables locales aumenta cada vez que se profundiza en la recursión. Así, con Factorial (30) se llegaría a tener en un momento dado 30 variables N ocupando espacio de memoria. Por ello, al escribir un procedimiento semejante debe comprobarse que no se va a seguir llamando a sí mismo indefinidamente. En el caso de Factorial esto está garantizado por la instrucción IF.

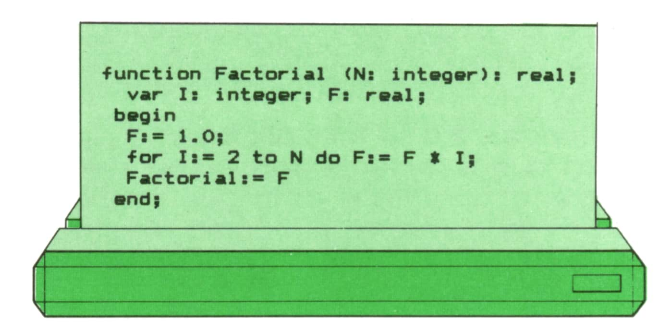
Hay muchos problemas de programa-

ción cuya respuesta se plantea de forma recursiva y ahí el empleo de procedimientos recursivos simplifica enormemente su solución; ejemplos típicos son los programas compiladores y los de gestión de bases de datos. Sin embargo, debido a la mayor cantidad de memoria que emplean y a la pérdida de tiempo que supone reservar y liberar memoria cada vez, deben utilizarse con precaución y sólo en los casos en que una programación no recursiva sea poco adecuada o impracticable.

La función factorial se puede calcular de manera no recursiva mucho más eficientemente utilizando un bucle para multiplicar N por N-1 por N-2... por 1:

«Calcular el factorial del parámetro»:
1 — Guardar 1.0 en la variable F.
1 — Para I variando su valor desde 2 hasta el parámetro hacer:
— Guardar en F su anterior valor multiplicado por el valor de I.
3 — Devolver el valor de F.

En el programa bastaría con sustituir la parte de la función por:



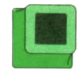
Este método de calcular el factorial de un número es mucho más rápido y necesita menos memoria que el recursivo.



Nota para curiosos

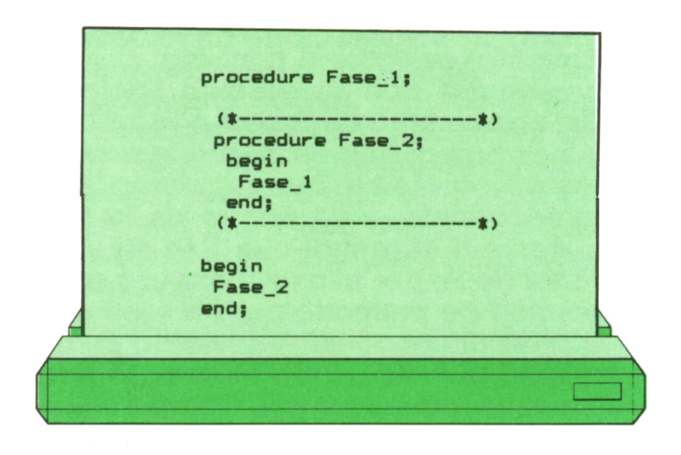
Aunque el BASIC, en principio, no permite programar algoritmos recursivos, con la mayoría de las versiones existentes sí es posible hacerlo simulando con tablas y punteros lo que, en lenguaje informático, se denomina «pila» para guardar ahí el equivalente a las variables locales del PASCAL. El caso concreto del cálculo recursivo de la función factorial es tan sencillo que ni siquiera hace falta eso, como se ve en el siguiente programa BASIC, más o menos equivalente al último ejemplo:

```
10 CLS
20 INPUT "Número"; N
30 GOSUB 100
40 PRINT N; "! = ";F
50 PRINT
60 GOTO 20
70 REM
80 REM ====================
90 REM
100 IF N=1 THEN F=1: RETURN
110 N=N-1
120 GOSUB 100
130 N=N+1
140 F=N*F
150 RETURN
```



Declaración anticipada de procedimientos

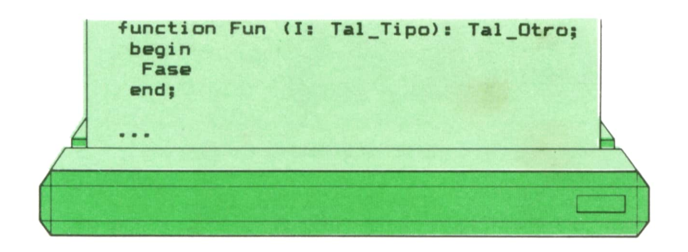
A veces se dan recursiones «a varias bandas» que pueden no notarse a primera vista, por ejemplo:



al ejecutarse Fase-1, éste utiliza a Fase-2, que a su vez utiliza a Fase-1, etc. (Fase-2 está dentro del inmediato poseedor de Fase-1).

Ligeramente distinto sería este otro caso de «dos bandas»:

```
...
procedure Fase;
 begin
  A:= Fun (Dato1)
 end;
```



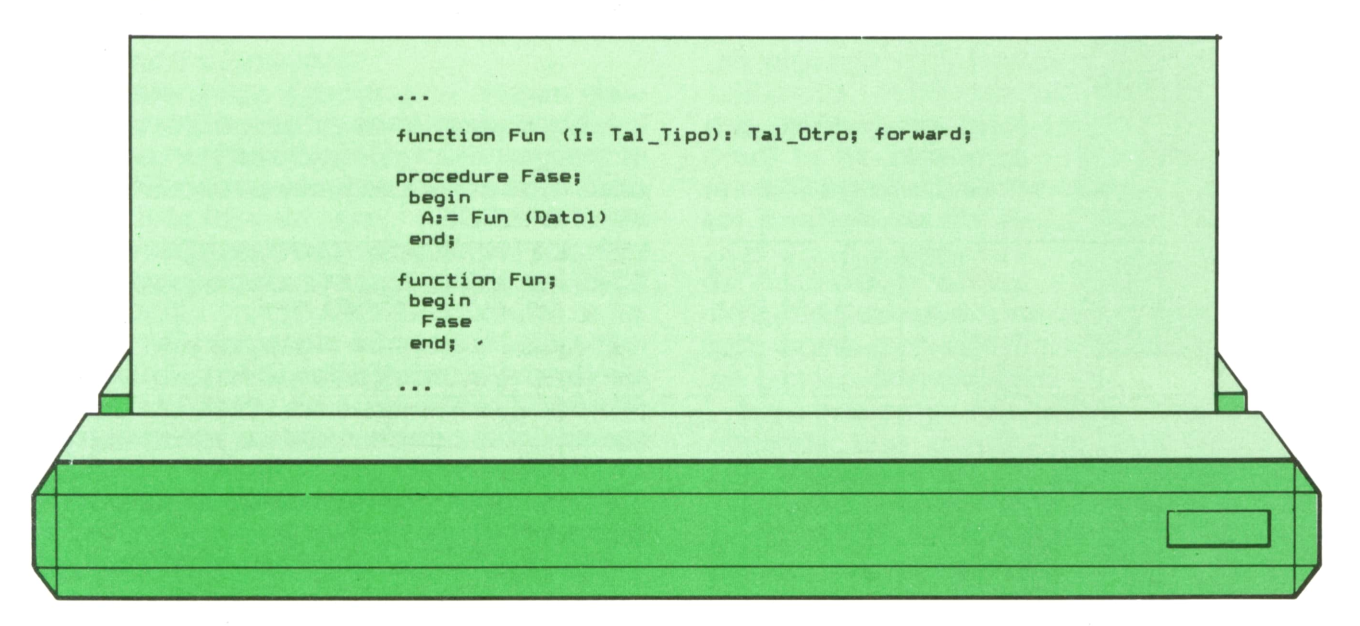
```
function Fun (I: Tal_Tipo): Tal_Otro;
 begin
  Fase
 end;

...
```

En principio, no sería posible compilar este programa, pues sólo se puede utilizar Fun desde detrás de su cabecera, y si se permutasen las posiciones de los subprogramas el problema se plantearía con Fase.

El PASCAL permite superar este problema mediante la descripción anticipada de la cabecera de un procedimiento o función. Para ello, en un lugar suficientemente adelantado del programa se escribe la cabecera completa y la palabra reservada FORWARD («adelante») separadas por un punto y coma.

Posteriormente, y ya donde esté escrito el procedimiento, se repite la cabecera sin poner la descripción de la lista de parámetros (si los hay) ni el tipo (si es una función):



```
...

function Fun (I: Tal_Tipo): Tal_Otro; forward;

procedure Fase;
 begin
  A:= Fun (Dato1)
 end;

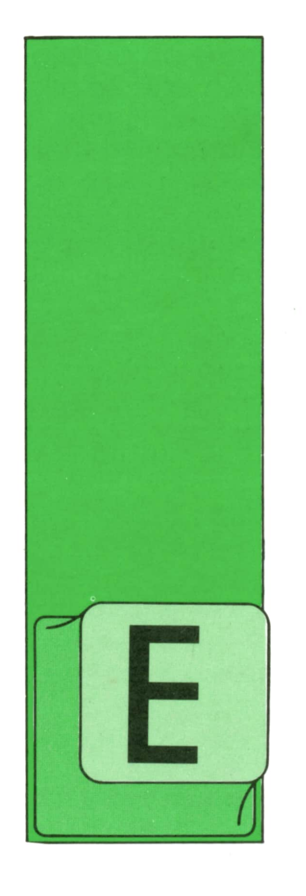
function Fun;
 begin
  Fase
 end;

...
```

De esta manera, cuando el PASCAL encuentre la llamada a Fun dentro del procedimiento Fase, ya tiene todo lo que hace falta saber desde fuera para utilizar la función.

OTROS LENGUAJES

DESCRIPCION DE CAMPOS



EN el programa que apareció en el tomo anterior se puede ver cómo se definen los campos en COBOL. Dicha definición se hace en la DATA DIVISION, que tiene dos secciones:

— FILE SECTION.

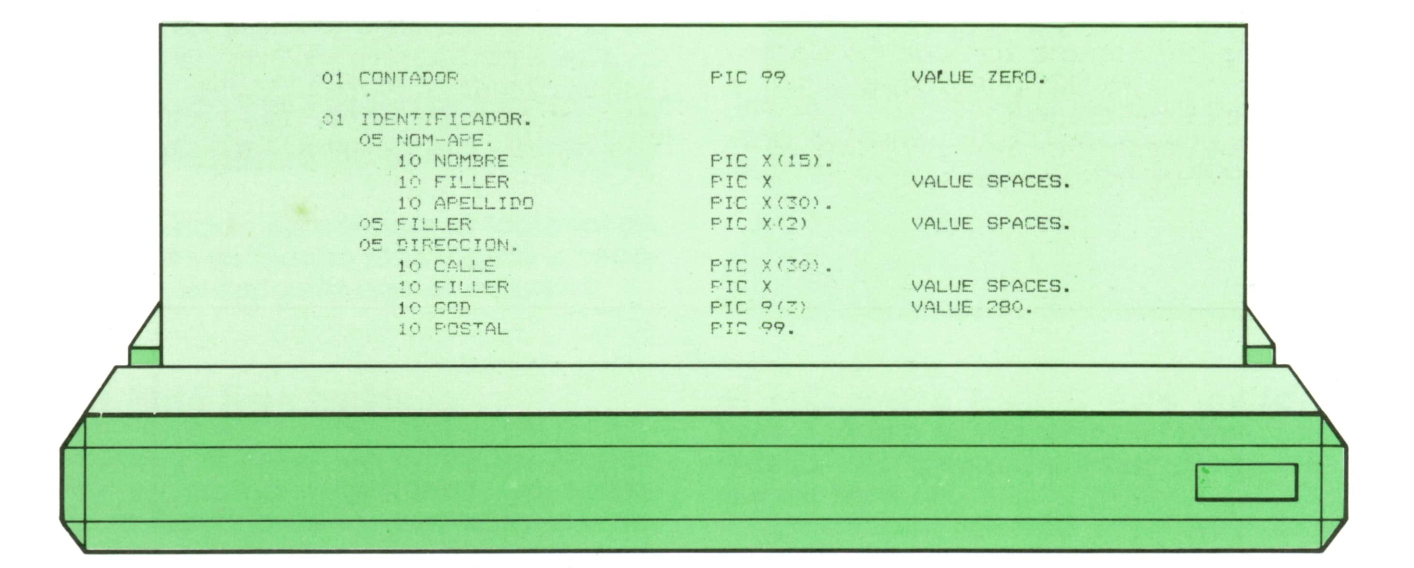
— WORKING-STORAGE SECTION.

Ambas secciones deben colocarse en el margen A. La primera sección se explicará más adelante junto con la ENVIRONMENT DIVISION.

Todos los campos de trabajo que se utilicen en un programa deben aparecer en la WORKING-STORAGE.

El COBOL permite describir estructuras de datos jerarquizadas. Esta jerarquía se fija con los **números de nivel.** Los números que se pueden utilizar están dentro del rango 01-49.

Un ejemplo de la descripción de campos es el siguiente:



El primer campo que se describe se llama CONTADOR. Todo campo debe tener un número de nivel y si no se subdivide en otros se dice que es un campo individual y tendrá como número 01.

Es obligatorio poner los números de nivel 01 en el margen A, pero el resto de niveles y nombres de campo deben ir en el margen B.

El nombre de los campos puede tener como máximo treinta caracteres, debe comenzar con un carácter alfabético (A-Z) y el resto pueden ser dígitos, letras y guiones.

Es conveniente que el programador dé a los campos nombres significativos, que mejoren la legibilidad del programa.

Es importante también resaltar que ningún nombre dado por el programador debe coincidir con una palabra reservada. Palabras reservadas son todas aquellas que son propias del lenguaje COBOL: PROCEDURE, FILE, ADD...

El siguiente campo es IDENTIFICADOR. Es un campo compuesto.

Los diferentes subcampos deben definirse con números de nivel mayores. En el ejemplo, estos subcampos se han descrito con los números 05 y 10, pero podían haber sido 02 y 03, etc.

NOM-APE es un subcampo que a su vez comprende otros tres: NOMBRE, FILLER y APELLIDOS.

Si en un programa se quisiera acceder sólo a los apellidos se haría referencia al campo APELLIDOS. Si se utiliza NOM-APE, se referencia el nombre y apellidos completos.

Con los números de nivel se puede obtener cualquier estructura de datos, estableciendo todas las subdivisiones precisas.

Detrás del nombre del campo aparecerá la cláusula PIC, que define el tipo y la longitud del campo.

Los dos caracteres más utilizados en la definición de los campos son:

— 9, indica que el campo es numérico. Almacena un dígito.

— X, el campo es alfanumérico, puede contener letras y números.

La longitud se determina por el número de 9's o X's que aparecen o por un número encerrado entre paréntesis.

CONTADOR tiene una longitud de 2, APELLIDOS 30 y DIRECCION 36 (que es la suma de los subcampos que la forman).

Los campos pueden inicializarse en la WORKING-STORAGE con la cláusula VALUE.

Para simplicar esta operación existen ya algunas constantes con valores predefinidos, llamadas **constantes figurativas.** Algunas de ellas son:

— ZERO: Su valor es cero.

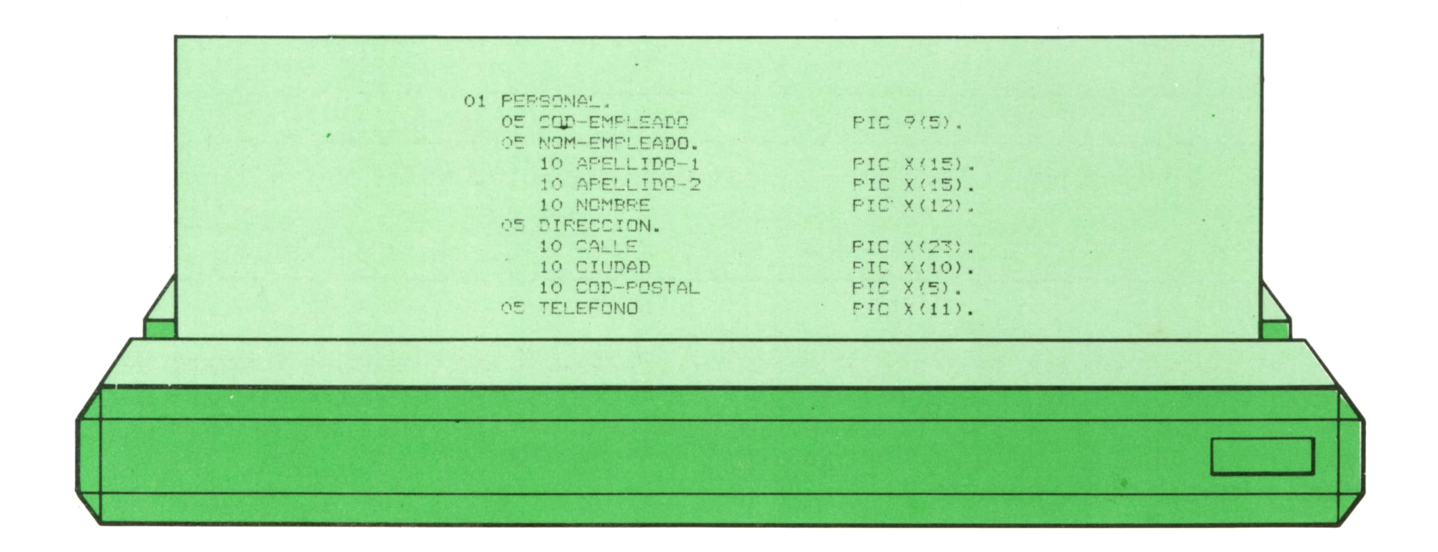
— SPACES: Inicializa el campo con espacios en blanco.

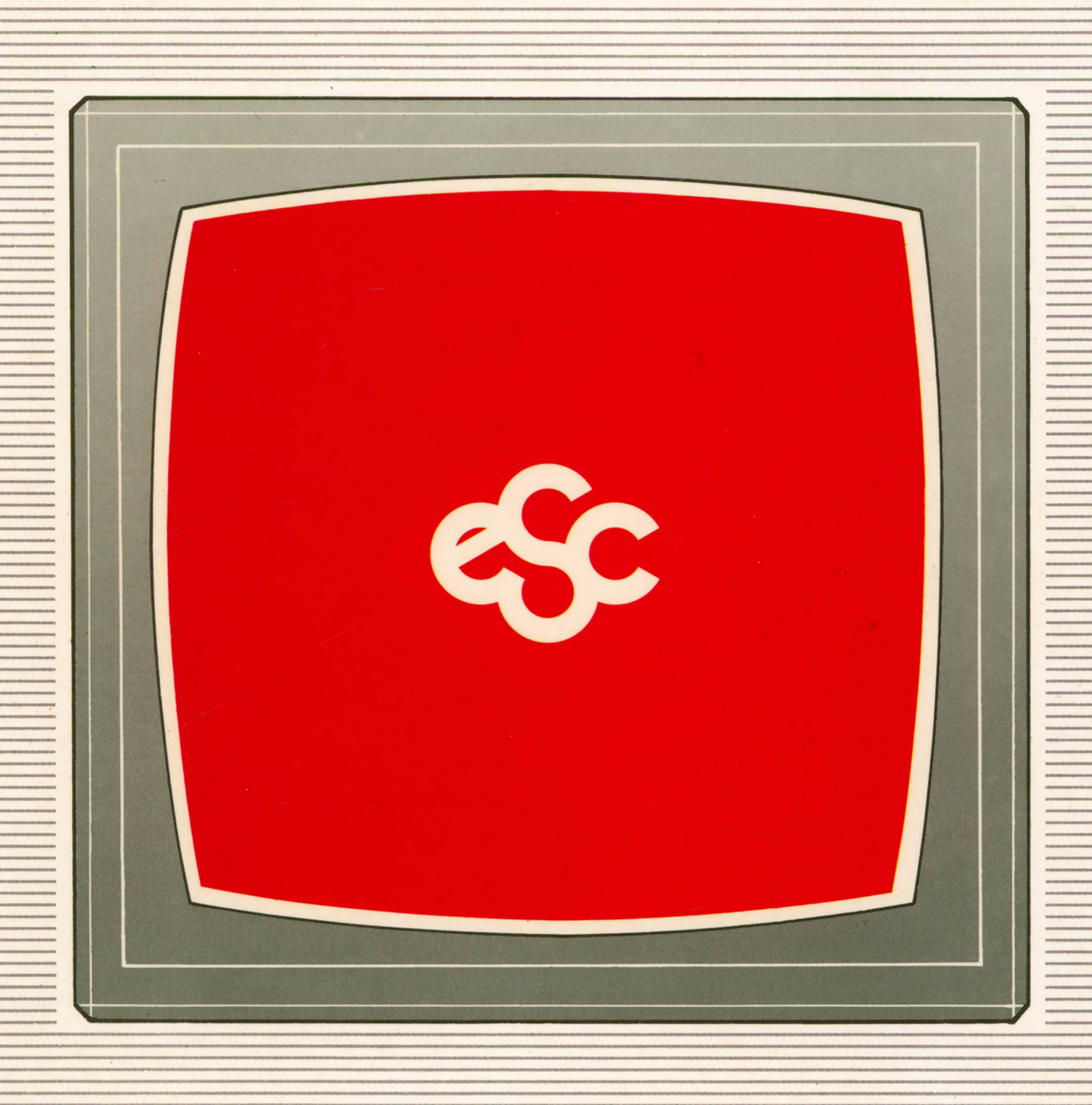
Por tanto, el valor inicial de CONTADOR es cero, el de los campos FILLER, espacios en blanco y el de COD 80.

Para inicializar este último campo se ha utilizado una constante numérica y no una figurativa.

En el ejemplo se pueden ver tres campos con el mismo nombre: FILLER. Este es común para todos los campos que no van a ser utilizados a lo largo del programa. De esta manera se evita el programador el crear nuevos nombres.

A continuación se muestra la composición del registro de un fichero de personal de una empresa.





▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼